z/OS

IBM z/OS Management Facility Programming Guide

Version 2 Release 1

SA32-1066-08

Note
Before using this information and the product it supports, read the information in "Notices" on page 761.

This edition applies to Version 2 Release 1 of IBM z/OS Management Facility (product number 5610-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this document

This document is intended to help you write programs that use the services and facilities of IBM® z/OS® Management Facility (z/OSMF).

The programming interfaces in z/OSMF include the following.

- Representational State Transfer (REST) services for working with z/OS and z/OSMF.
- XML schema for creating a workflow definition for performing activities on a z/OS system, such as configuring a component or product.
- Services for creating plug-ins that add installation-specific function to z/OSMF.

Who should use this document

This information is intended for the programmer responsible for writing programs that use the z/OSMF infrastructure. Such activities, include, for example:

- Using an application programming interface (API) as a client to obtain information about a batch job on z/OS
- Developing a JavaScript program that includes a graphical user interface (GUI).

This document assumes that you are familiar with the z/OS operating system and its accompanying products.

Where to find more information

For an overview of the information associated with z/OS, see *z/OS Information Roadmap*.

z/OSMF home page

Visit the z/OSMF home page at http://www.ibm.com/systems/z/os/zos/zosmf/.

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your web browser to the following web site, which is available to all users (no login required): http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp.

How to send your comments to IBM

We appreciate your input on this documentation. Please provide us with any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

Use one of the following methods to send your comments:

Important: If your comment regards a technical problem, see instead "If you have a technical problem."

- Send an email to mhvrcfs@us.ibm.com.
- Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).

Include the following information:

- Your name and address
- Your email address
- Your phone or fax number
- The publication title and order number:
  IBM z/OSMF Programming Guide
  SA32-1066-05
- The topic and page number or URL of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one or more of the following actions:

- Visit the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-08

This document contains information that was previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-07, which supported IBM z/OS Management Facility Version 2 Release 1. This document contains new or revised information.

New information

When you install the January 2017 functional updates, new functions are added to z/OSMF V2R1.

- New REST APIs that can be used to provision z/OS software in support of IBM Cloud Provisioning and Management for z/OS are added. For an introduction, see "Cloud provisioning services" on page 46. For more information about the APIs, see the following:
  - Resource management REST APIs, for managing domains and tenants. For more information, see "Resource management services" on page 70.
  - Resource Pool REST APIs, for managing network resource pools. For more information, see "Resource pool services" on page 49.
  - Software services template REST APIs, for managing templates that are used to provision software. For more information, see "Software services template services" on page 82 and "Published software service template services" on page 138.
  - Software services instance REST APIs, that are used to manage software services instances that represent provisioned software. For more information, see "Software services instance services" on page 157.
  - Software service instance name (SSIN) APIs. For more information, see "Software service instance name services" on page 204.
  - Workload management resource APIs. For more information, see "WLM resource pooling services" on page 351.

  A new topic has been added for software providers, Chapter 4, "Preparing software for cloud provisioning," on page 743.
- New REST APIs that can be used to work with archived workflows, as follows:
  - "Archive a workflow instance" on page 571
  - "List the archived workflows for a system" on page 573
  - "Get the properties of an archived workflow" on page 576
  - "Delete an archived workflow" on page 588.
- System variable REST APIs. For more information, see "z/OSMF system variable services" on page 513.
- The Workflows task of z/OSMF now includes an editor for workflows. You can use the Workflow Editor to view, create, and modify workflow definitions. The Workflow Editor provides a visual framework for working with the elements of a workflow definition. To get started with the Workflow Editor task, in the navigation area, select **Workflow Editor**. For more information, see "Workflow Editor task in z/OSMF" on page 594.

Changed information

When you install the January 2017 functional updates, changes are made to z/OSMF V2R1.

- Enhancements are made to the Workflows XML schema that is supplied with z/OSMF. This file provides the XML syntax and rules for creating a workflow definition. The enhancements allow workflow authors to create workflow *REST steps*, which are steps that issue Representational State Transfer (REST) requests, such as GET or PUT.

  For information about REST steps and the schema elements that you can use to create them, see "REST steps" on page 618.
- Enhancements are made to the z/OS Workflows REST interface services API. This programming interface allows the caller to create and manage z/OSMF workflows on a z/OS system. The enhancements support the use of REST steps, as follows:
  - The service that is described in "Get the properties of a workflow" on page 533 is updated to return the properties of a step that issues a REST request. The properties are included in the step-info JSON object that is returned to the requester.
  - The service that is described in "Retrieve a workflow definition" on page 560 is updated to return the schema elements for the REST steps in a workflow.
- New terms are introduced. To help you distinguish REST steps from other types of steps, the terms *template step* and *REST step* are introduced in this publication, as follows:
  - *Template step* is used to indicate a step that runs a program, such as a JCL job, REXX exec, or UNIX shell script
  - *REST step* is used to indicate a step that issues a REST request, such as GET or PUT.
- To give workflow authors more control over how variables are used, the following options are added to the Workflows XML schema:
  - New attribute `visibility`, which is added to the element variable (`<variable>`), specifies whether a variable is intended for public or private use. See "Specifying the variable element and its attributes" on page 633.
  - New element atCreate (`<atCreate>`) can be used to set attributes for a variable across workflows. See "Using the element atCreate to qualify a variable definition" on page 636.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-07

This document contains information that was previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-06, which supported IBM z/OS Management Facility Version 2 Release 1. This document contains new or revised information.

New information

New z/OSMF Representational State Transfer (REST) APIs for z/OS console services. See "z/OS console services" on page 362.

New REST APIs that can be used to work with archived workflows, as follows:

- "Archive a workflow instance" on page 571
- "List the archived workflows for a system" on page 573
- "Get the properties of an archived workflow" on page 576
- "Delete an archived workflow" on page 588

A workflow author can optionally include a feedback form with customized questions for the step owner to answer at the conclusion of a step. Such feedback can be useful for determining the effectiveness of a workflow design, or collecting user requirements for future enhancements to a workflow. For information, see "Collecting user feedback" on page 606.

| New and changed REST APIs for working with z/OS data sets, including z/OS UNIX Systems Services
| files. See “z/OS data set and file REST interface” on page 380.

Moved information

Appendix B, “Creating product information files for the Software Management task,” on page 749 is moved to this document. Previously, this information appeared in *IBM z/OS Management Facility Configuration Guide*.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-06

This document contains information that was previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-05, which supported IBM z/OS Management Facility Version 2 Release 1. This document contains new or revised information.

New information

New z/OSMF Representational State Transfer (REST) APIs for notifications. See “Notification services” on page 258.

Enhancements are made to the z/OSMF workflow services API, as follows:

- *Access type* is a new security control for workflows, which is used to control what the user is permitted to see and modify in various sections of the workflow. The access type is specified by the workflow owner at workflow creation time.

  The following services are updated to use the access type:
  - “Create a workflow” on page 528
  - “List the workflows for a system or sysplex” on page 549
  - “Get the properties of a workflow” on page 533

Changes made in z/OSMF Version 2 Release 1, SA32-1066-05

This document contains information that was previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-04, which supported IBM z/OS Management Facility Version 2 Release 1. This document contains new or revised information.

New function is available for z/OSMF V2R1 when you install the PTFs for APAR PI54286 and the corequisite APARs. APAR PI54286 and the corequisite APARs provide function that was originally provided with z/OSMF V2R2. For instructions on installing this service on a z/OSMF V2R1 system, check the++HOLD actions for the associated PTFs.

Additional function is available with the March 2016 updates.

New information

The multisystem routing services are enhanced to allow a calling application to:
- Retrieve data from all the systems in a sysplex or CPC.
- Update data for all the systems in a sysplex or CPC.
- Delete data from all the systems in a sysplex or CPC.

For more information, see “Multisystem routing services” on page 226.

New services are added to the software management API to allow a calling application to:
- List the data sets included in a software instance.
- Modify the properties of a software instance.
- Load the products, features, and FMIDs for a software instance.

- Delete a software instance from z/OSMF.

For more information, see "Software management services" on page 270.

The topology services adds support for central processor complexes (CPCs) so that you can retrieve a list of the systems included in a CPC. For more information, see "List the systems included in a CPC" on page 328.

New REST services are added to allow a calling application to:
- Create a sequential and partitioned data set.
- Create a UNIX file or directory.
- Delete a sequential or partitioned data set.
- Delete a partitioned data set member.
- Delete a UNIX file or directory.
- Mount a UNIX file system.
- Unmount a UNIX file system.

For more information, see "z/OS data set and file REST interface" on page 380.

New elements in the Workflows XML schema provide additional capabilities, as follows:
- You can enable a workflow for future upgrades by adding the `<preserveOptions>` element and associated elements to the workflow definition. If so, the workflow owner can use the **Create New Based on Existing** action that is provided in the Workflows task to upgrade a workflow definition to a new level without losing data or having to start over again by creating a new instance. For information, see "Enabling a workflow definition file for future upgrades" on page 603.
- Workflows users can add their own steps to a workflow by using the **Update Workflow Steps** action that is provided in the Workflows task. Users can also modify or delete the user-supplied steps, as needed. For more information, see the Workflows task online help.
- You can use substitution variables in the file path of an external file, such as a step template. For information, see "Using variable substitution in the workflow definition file path" on page 598.
- You can use shorter names for instance variables; see "Simplified instance variable format in substitution and conditions" on page 638.
- You can specify a workflow scope of *none* to cause a new instance of a called workflow to always be created. See "Coordinating workflow-to-workflow actions" on page 621.

Changed information

For workflow authors, a requirement is removed. It is no longer necessary to specify the variable value element (`<variableValue>`) for instance variables that are referenced in a step, if the variables are used read-only. The `<variableValue>` element is still required for any variables that might be modified in the step. When `<variableValue>` is specified for a variable, the Workflows task displays a wizard to guide the user through entering values for the variables when performing the step.

Previously, it was a requirement that you specify `<variableValue>` and its attributes for all instance variables that are referenced in a step, even variables that would not be modified. This requirement is removed to save programming time and simplify the display of variables in the Workflows task.

z/OSMF REST APIs for Files and Data sets have been modified. See "z/OS data set and file REST interface" on page 380. API updates include:
- List the data sets on a system.
- List the members of a data set.
- List the files and directories of a UNIX file path.
- Retrieve the contents of a data set or member.
- Retrieve the contents of a UNIX file.
- Write to a data set or member.
- Write to a UNIX file.

- JSON document specifications for z/OS data set and file REST interface requests.
- Error reporting categories.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-04

This document contains information that was previously presented in *IBM z/OS Management Facility Programming Guide*, SA32-1066-03, which supported IBM z/OS Management Facility Version 2 Release 1. This document contains new or revised information for APAR PI40364.

New information

New functionality is available for z/OSMF V2R1 when you install APAR PI40364. For instructions on installing this service on your system, check the ++HOLD actions for the associated program temporary fix (PTF).

New Representational State Transfer (REST) services are added to route requests and responses between the client-side and server-side code for any z/OSMF plug-ins you created where the server-side code is hosted on an application server (referred to as the *target application server*) other than the z/OSMF server. The new services allow a caller to:
- Retrieve data from the target application server.
- Update data on the target application server.
- Delete data from the target application server.

For more information, see "Application server routing services" on page 31.

New elements in the Workflows XML schema provide additional capabilities, as follows:
- With the exposeToUser (`<exposeToUser>`) element, you can include variables in the List variables for substitution window of the Workflows task. Doing so allows users to select the variables for use in the JOB statement for a job.
- With the predefinedVariable (`<predefinedVariable>`) element, you can specify one or more predefined variables for a step.

For more information, see "Using variables in template steps" on page 616.

With the _output variable, you can create step-specific values in a step that creates an output file. For more information, see "Create step-specific values with the _output variable" on page 640.

Changed information

When using the multisystem routing services, if the value for a parameter contains a number sign (#), encode the number sign as *%23*. Otherwise, everything following the number sign will be omitted from the request. For example, if the target is *System#1*, specify *System%231*.

For more information, see "Multisystem routing services" on page 226.

Workflow authors can include larger programs of up to ten thousand (10000) lines of code in steps that run JCL jobs, REXX execs, or UNIX shell scripts. Previously, workflow steps were limited to programs of one thousand (1000) lines of code.

For more information, see "Template steps" on page 611.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-03

This document contains information that was previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-02, which supported IBM z/OS Management Facility Version 2 Release 1. This document contains new or revised information for APAR PI32148 and the corequisite APARs.

New information

New functionality is available for z/OSMF V2R1 when you install APAR PI32148 and the corequisite APARs. For instructions on installing this service on your system, check the ++HOLD actions for the associated program temporary fix (PTF).

New Representational State Transfer (REST) services are added to enable communication and data transfer between systems within your enterprise. The new services allow a caller to:
- Retrieve data from one system, a list of systems, or all the systems in a group
- Update data for one system, a list of systems, or all the systems in a group
- Delete data from one system, a list of systems, or all the systems in a group
- Authenticate with a secondary z/OSMF instance.

For more information, see "Multisystem routing services" on page 226.

New REST services are added to allow a client application to interact with the z/OSMF Software Management task. The new services allow a caller to:
- List the software instances that are defined to z/OSMF
- Retrieve the properties of a software instance
- Add a new software instance.

For more information, see "Software management services" on page 270.

New REST services are added for working with the groups, sysplexes, and systems that are defined to z/OSMF. The new services allow a caller to:
- List the systems that are defined to z/OSMF
- List the groups that are defined to z/OSMF
- List the systems included in a group
- List the sysplexes that are defined to z/OSMF
- List the systems included in a sysplex.

For more information, see "Topology services" on page 313.

New REST services are added for creating and managing a workflow on a z/OS system. The new services allow a caller to:
- Create a workflow
- Get the properties of a workflow
- List the workflows for a system or sysplex
- Start a workflow
- Cancel a workflow
- Delete a workflow
- Retrieve a workflow definition.

For more information, see "z/OSMF workflow services" on page 525.

A workflow step can be designed to call another workflow for execution. Here, the target workflow is referred to as a *called workflow*. Conceptually, a called workflow is simply another step in the workflow that calls it. For more information, see "Calling steps" on page 621.

Enhancements are made to the Workflows task schema that is supplied with z/OSMF. This file provides the XML syntax and rules for creating a workflow definition. The Workflows task schema is described in "Workflow XML reference" on page 643.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-02

This document contains information previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-01, which supported IBM z/OS Management Facility Version 2 Release 1.

This document contains new or revised information for APAR PI20091 and the corequisite APARs.

New information

New functionality is available for z/OSMF V2R1 when you install APAR PI20091 and its corequisite APARs. For instructions on installing this service on your system, check the ++HOLD actions for the associated program temporary fix (PTF).

New Representational State Transfer (REST) services are added to the z/OS data set and file REST interface API. This programming interface allows an HTTP client application to work with data sets and UNIX files on the z/OSMF host system. The new services allow a caller to:
- List the members of a z/OS partitioned data set (PDS or PDSE)
- Retrieve the contents of a sequential data set, or a member of a PDS or PDSE
- Retrieve the contents of a z/OS UNIX file
- Write data to a sequential data set or a member of a PDS or PDSE
- Write data to a z/OS UNIX file.

For more information, see "z/OS data set and file REST interface" on page 380.

A new REST service is added to allow a caller to obtain information about z/OSMF on the z/OS host system, such as version, release, and service level. With this information, a program can determine which z/OSMF plug-ins and core functions are available for use on a given system. For information, see "z/OSMF information retrieval service" on page 509.

Changed information

Enhancements are made to the z/OS jobs REST interface services API. This programming interface allows an HTTP client application to work with batch jobs on the z/OSMF host system. The enhancements are as follows:
- Some REST services can be run synchronously, if coded to use the latest version of the service. The enhanced services are:
  - Hold a job
  - Release a job
  - Change the job class
  - Cancel a job
  - Delete a job.

  Previously, these services were limited to running asynchronously only.

  Synchronous processing is supported for JES2 subsystems only. When the target subsystem is JES3, a synchronous request is ignored and the service is performed asynchronously, as in prior releases.
- Related to this change, the enhanced services, if run synchronously, no longer require that your user ID be authorized to the Common Information Model (CIM) server and permitted to the JES2-JES3Jobs CIM provider.

For more information, see "z/OS jobs REST interface" on page 461.

The tracing property for JES activities for your programs is changed to **izurestjobs.logging**. Previously, this setting was **izurestjobs.jessymbols.logging**. For more information, see "Tracing the JES related activities for your programs" on page 747.

Enhancements are made to the Workflows task schema that is supplied with z/OSMF. This file provides the XML syntax and rules for creating a workflow definition. The Workflows task schema is enhanced, as follows:

- A workflow step can be designed to save its output in a separate file called the *output file*. On completion of the step, the contents of the output file become available for use by subsequent steps in the workflow instance, or by other workflow instances.
- A workflow step can be designed to be performed conditionally, based on whether a logical condition is satisfied on the z/OS system. For example, a conditional step might become eligible to be performed if a job run by another step ends with a particular return code.

For more information, see Chapter 2, "Creating workflow definitions for z/OS," on page 591.

If you are creating your own plug-ins and are supplying context-sensitive documentation, the URL for linking the help content to the user interface has changed to:

```
https://<host>:<port>/<context-root>/helps/SSB2H8_2.1.0/<help-plugin-name>/<help-topic>
```

For information, see "Adding links to help plug-ins" on page 733.

Changes made in z/OSMF Version 2 Release 1, SA32-1066-01

This document contains information previously presented in *IBM z/OS Management Facility Programming*, SA32-1066-00, which supported IBM z/OS Management Facility Version 2 Release 1.

This document contains new or revised information for APAR PM98630 and its corequisite APARs.

New information

New functionality is available for z/OSMF V2R1 when you install APAR PM98630 and its corequisite APARs. For instructions on installing this service on a z/OSMF V2R1 system, check the ++HOLD actions for the associated program temporary fix (PTF).

Three new application programming interfaces (APIs) are available:

- Data persistence services, which allows an HTTP client application to perform the following actions:
  - Persist user-specific and global application data.
  - Retrieve user-specific and global application data.
  - Delete user-specific and global application data.

  For information, see "Data persistence services" on page 215.
- TSO/E address space services, which allows an HTTP client application to work with TSO/E address spaces on the z/OSMF host system. The supported operations are:
  - Start or reconnect to a TSO/E address space.
  - Start an application on a TSO/E address space.
  - Receive messages from a TSO/E address space.
  - Receive messages from an application running in a TSO/E address space.
  - Send messages to a TSO/E address space.
  - Send messages to an application running in a TSO/E address space.
  - Ping a TSO/E address space.
  - End a TSO/E address space.

  For information, see "TSO/E address space services" on page 331.
- z/OS data set and file REST interface, which allows an HTTP client application to work with z/OS data sets and UNIX files on the z/OSMF host system. The supported operations are:
  - List data set names
  - List z/OS UNIX directories or files.

  For information, see "z/OS data set and file REST interface" on page 380.

To enable the z/OS data set and file REST interface services, IBM supplies a default procedure in your order, which you must install prior to configuring z/OSMF. For information, see *IBM z/OS Management Facility Configuration Guide*.

If your installation requires function that is not provided in z/OSMF or in another web-based application, you can create your own plug-in and import it into z/OSMF. For information, see Chapter 3, "Creating your own z/OSMF plug-ins," on page 683.

The z/OS jobs REST interface services API is enhanced, as follows:

- A calling application can submit a job to run on a secondary JES2 system. In previous releases, jobs could be submitted to the primary JES2 system only. See "Submit a job" on page 476.
- New services are added to allow a calling application to:
  - Hold a job to make it ineligible for processing; see "Hold a job" on page 481
  - Release a job that has been held so that it can be selected for processing; see "Release a job" on page 484.

For more information about z/OS jobs REST interface services, see "z/OS jobs REST interface" on page 461.

To help you create a workflow definition, z/OSMF has added a number of functional enhancements to the Workflows task schema. With these enhancements, you can:

- Pre-specify the user input values for a workflow through a property file called the *workflow variable input file*. With this file, you can lessen or eliminate the need for the user to specify inputs manually when running a workflow. More information is provided in "Providing a workflow variable input file" on page 639.
- Create workflows with *automated steps*, that is, steps that run with little or no user interaction when the required user input values are satisfied beforehand, such as through a workflow variable input file. More information is provided in "Automated steps" on page 627.

For z/OSMF Version 2 Release 1, SA32-1066-00

This document is new for z/OSMF Version 2 Release 1. It contains information that was previously presented in *IBM z/OS Management Facility Configuration Guide*, SA38-0652-07.

New information

In this release, you can create a workflow definition for performing activities on a z/OS system, such as configuring a component or product. An active workflow is created from the workflow definition when a user imports the workflow definition into the Workflows task of z/OSMF.

To help you get started with creating workflow definitions, z/OSMF includes XML samples for your reference. For more information, including concepts and a description of the basic elements of a workflow definition file, see Chapter 2, "Creating workflow definitions for z/OS," on page 591.

In this release, the z/OS jobs REST interface services are enhanced, as follows:

- You can use a job correlator as an alternative to specifying a job name and job ID combination. A job correlator provides you with a means to query a job in the system and track it through execution. This function is available for JES2 systems only; it is not available for JES3 systems.
- The service described in "Submit a job" on page 476 is enhanced to allow the specification of one or more JCL symbols. This function is available for JES2 systems only; it is not available for JES3 systems.
- You can use the asynchronous job notifications function of z/OSMF to allow your HTTP applications to be notified when submitted jobs complete. With this function, a program that submits a job through the z/OS jobs REST interface services POST method specifies a URL when submitting the job. When the job ends, z/OSMF returns an HTTP message to the URL location, indicating the job completion status.

For information about the z/OS jobs REST interface services, see "z/OS jobs REST interface" on page 461. This function is available for both JES2 systems and JES3 systems. For information about the asynchronous job notifications function, see the topic on configuring your system for asynchronous job notifications in *IBM z/OS Management Facility Configuration Guide*, SA38-0657.

Information applicable to all releases

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line in the margin by the change.

The *Readers' Comments - We'd Like to Hear from You* section at the end of this publication has been replaced with a new section "How to send your comments to IBM" on page xxi. The hardcopy mail-in form has been replaced with a page that provides information appropriate for submitting comments to IBM.

Chapter 1. Using the z/OSMF REST services

z/OSMF supports the use of Representational State Transfer (REST) APIs, which are public APIs that your application can use to work with system resources and extract system data. As with implementations of REST services on other platforms, the z/OSMF APIs allow for easy-to-use services that are language- and platform-independent, stateless, scalable, and easily parsed.

Processing overview

The z/OSMF REST services can be invoked by any hypertext transfer protocol (HTTP) client application, running on the z/OS local system or a remote system. The services support requests in either of the following protocols: HTTP/1.0 or HTTP/1.1.

Conceptually, your application (the client) issues requests to the target system (z/OS) in the form of request messages. The product name is z/OS. Each request message consists of a request line, optionally followed by request headers (HTTP headers), an empty line, and an optional message body. The request line includes the HTTP method, such as GET, a Universal Resource Locator (URL) and, where appropriate, parameters that further qualify the request. The URL is required and must be URI-encoded as specified in RFC 2396. For more information about RFC 2396, see the Uniform Resource Identifiers (URI): Generic Syntax web page.

If the API determines that the request is valid, it performs the requested service. After performing the service, the API creates an HTTP response. If the request is successful, this response takes the form of an HTTP 200 (OK) response and, if applicable, an object that contains a result set. Depending on which service was requested, the result set might be returned in a format that requires parsing by your program, for example, a JavaScript Object Notation (JSON) object. In other cases, results might be returned in another format, such as plain text or binary data. If the request is not successful, the response consists of a non-OK HTTP response code with details of the error provided in the form of a JSON object.

It is assumed that users of these services are familiar with the JSON standard and coding practices. The following references provide more helpful information:

- Hypertext Transfer Protocol 1.1: http://www.w3.org/Protocols/
- Multipurpose Internet Mail Extensions (MIME) media types: http://www.iana.org/assignments/media-types/index.html
- Introducing JSON: http://www.json.org.

Authenticating to z/OSMF

Your client program must authenticate to z/OSMF on the target z/OS system. Typically, authentication is done through the HTTP header included in each client request, which is referred to as a *stateless* technique.

You might also consider using a single sign-on or *stateful* technique, such as the following:

1. On the first request to the server, the client request includes a basic HTTP authentication header that contains a valid user ID and password. The header property value pair should look like this:

   ```
   'Authorization':'Basic <encoding of userid:password>'
   ```

   where `<encoding of userid:password>` is a base 64 encoding of `<userid>:<password>`.
2. On successful log-in, your application receives the following values in the response header:
   - HTTP status code 200

- Lightweight Third Party Access (LTPA) token, which contains the credentials for your application. In the case of z/OSMF, the token is a LtpaToken2 value, which supports strong encryption.

3. On subsequent requests, your application supplies the LTPA token for authentication with z/OSMF instead of the basic HTTP authorization header.

   You can provide the LTPA token through the Cookie header property, for example:

   ```
   'Cookie':'<ltpaToken2=<tokenvalue>'
   ```

   followed by the token, for example:

   ```
   'Cookie': 'LtpaToken2=IExabotu2sfNbJij6rajHJcFiDi
   1H0hml3yKvylwfJ4q8goCFEYH41FQNlAgdMMVP6/nVbH/IKw
   0l5b7ZqWuZ8ndOYcECAJg1ss2Vq4q21C1jLvVGTyNLk6rvbgs
   7oQWM98bSuAN1Qtvlrx9uZ8EY4GqqscaErQO9vrAhwgkcedWB
   jn2lLNjl+G8olJA4uB+Cv5XamrUvziY2pcbCKjFjNt5EQ97Nf2
   sBzvlanfrENhV9uOLRpw9DibrzKLh0R1fOrp5xySAe7Ery69
   eynt4ItaVWCcpt+CYHFbpHpW/C7INWHeNcaNktr0DBmHh6EWl; '
   ```

Supported HTTP versions

The z/OSMF REST interface services support requests in either of the following protocols: HTTP/1.0 or HTTP/1.1

Usage considerations for the z/OSMF REST services

Observe the following considerations when using the z/OSMF REST interfaces:

- As with any z/OSMF task or function, the REST services compete for z/OSMF resources with users of the z/OSMF web browser interface. Thus, concurrent high usage of the REST services can affect response time for users of the z/OSMF web browser interface.
- During periods of concurrent high usage of the REST services, an application can experience connection failures, such as connection refused, connection timed out, or connection reset. In these cases, the application should try the request again. The number of retry attempts needed will depend on how much work is being requested of the server. It might be necessary for your installation to modify the workload and reduce the arrival rate of requests.
- Some browsing environments do not support all of the HTTP methods, such as HTML 4 or XHTML 1, or might block applications from accessing response content having non-successful HTTP response status codes (4*nn* and 5*nn*). As a workaround, your application can use the following custom HTTP request headers:

  X-IBM-Requested-Method:
  GET, PUT, and DELETE requests can be "tunneled" through a **POST** method using this custom HTTP header.

  X-IBM-Bypass-Status:
  If set to true, all response status codes are set to 200, and the custom HTTP response header **X-IBM-Actual-Status** is included in the returned data. To determine the original status code, your application must check the **X-IBM-Actual-Status** header.

A note on the timestamp data type

The timestamp data type is used in the definition of some data models and notification message formats in the z/OSMF REST services APIs. Where it appears in this document, the timestamp data type is used to mean a non-negative Long integer quantity where the value represents a date and time expressed as the number of milliseconds since midnight on January 1, 1970 UTC.

Allowing cross-site access to REST services

The z/OSMF REST services can be accessed by an HTTP client application, or by a web application running on the host system. By default, z/OSMF blocks access attempts from web applications on remote systems. In such cases, the request is failed with an error message indicating that a cross-site request forgery (CSRF) was attempted.

Your installation can choose to allow cross-site access to REST services on the host system. When you have identified which sites are to be allowed, and which REST interfaces are to be made available for cross-site access, work with your security administrator to create the appropriate authorizations in your z/OS security product. This work involves defining generic or discrete profiles for the remote sites in the ZMFAPLA class, and permitting the profiles to the z/OSMF REST interfaces.

To define a profile for a remote site, use the following format:
`<SAF_PREFIX>.REST.<identifier>.<reversed-hostname>`

where:

- `<SAF_PREFIX>` is the SAF prefix for your z/OSMF configuration. By default, this is `IZUDFLT`.
- `REST.<identifier>` identifies the REST interface that is to be allowed for use by the remote site. Table 1 shows the identifiers for each of the z/OSMF interfaces. To indicate all REST interfaces, specify an asterisk (*) as the identifier.

Table 1. SAF identifiers for the z/OSMF REST interfaces

REST interface	Identifier
Application Linking Manager interface	APPLINK
Application server routing services	GATEWAY
Cloud provisioning services	PROVISIONING
Data persistence services	PERSIST
Multisystem routing services	GATEWAY
Software management services	SWMGMT
Topology services	SYSTEM
TSO/E address space services	TSO
z/OS data set and file REST interface	FILES
z/OS jobs REST interface	JOBS
z/OSMF workflow services	WORKFLOW

- `<reversed-hostname>` is the site's fully qualified domain name in reverse, for example, the domain `WWW.IBM.COM` would be specified as `COM.IBM.WWW`. Use uppercase letters for any alphabetic characters in the profile. Specify a domain name only; not the full URL. Omit the protocol (`HTTP://` or `HTTPS://`).

  If the site is known by an IP address, specify the IP address in reverse. For any IP addresses that you define, it is recommended that you create a discrete profile (without wildcards) for each address. Use a valid "dotted decimal" Internet Protocol version 4 (IPv4) address. IPv6 addresses and internationalized domain name (IDN) addresses are not supported.

  Though not recommended, you can use generic profiles with wildcard characters to allow access from multiple domains or sub-domains. You might do so temporarily to allow for internal testing of the REST interfaces across multiple sandbox systems.

For example, assume that your installation wants to allow a web application on the site "lab2.ibm.com" to send requests for z/OS jobs REST interface services on the site "lab1.ibm.com." To allow cross-site requests from the "lab2" site, your RACF security administrator would create the authorization on the "lab1.ibm.com site," as follows:

1. Create a profile for the "lab2.ibm.com" site in the ZMFAPLA class. In the profile, include the identifier to represent the resource (the REST interface), for example, JOBS to indicate the z/OS jobs REST interface. In RACF, enter the **RDEFINE** command, as follows:

   ```
   RDEFINE ZMFAPLA IZUDFLT.REST.JOBS.COM.IBM.LAB2 UACC(NONE)
   ```

   To allow this cross-site access for all of the REST interfaces, specify a wildcard (*) in place of a specific resource identifier. For example:

   ```
   RDEFINE ZMFAPLA IZUDFLT.REST.*.COM.IBM.LAB2 UACC(NONE)
   ```

2. Ensure that the z/OSMF server user ID (by default, IZUSVR) has READ access to the profile. In RACF, enter the **PERMIT** command, as follows:

   ```
   PERMIT IZUDFLT.REST.JOBS.COM.IBM.LAB2 CLASS(ZMFAPLA) ID(IZUSVR) ACCESS(READ)
   ```

3. Refresh the ZMFAPLA class. In RACF, enter the **SETROPTS** command, as follows:

   ```
   SETROPTS RACLIST(ZMFAPLA) REFRESH
   ```

Note: The settings you make here will not override any security functions in place in your browser. Consult your browser documentation for more details regarding cross-site scripting.

Where the z/OSMF REST services are described

The z/OSMF REST services are described in the sections that follow.

Application Linking Manager interface services

To perform traditional system management tasks in z/OS, you might interact with several different interfaces, such as the TSO command line, graphical user interfaces, and web-style interfaces. With the z/OSMF Application Linking Manager, it is possible to link or connect some of these tasks and external applications together for a smoother user experience.

The key components in the Application Linking Manager process include the:

- **Event requestor.** z/OSMF task or external application that requests the launch of a specific function within another task or external application
- **Event.** Action requested by the event requestor. It includes the type of event and the event parameters.
- **Event type.** Object that connects an event requestor to an event handler. It identifies the handlers that can process an event and the possible parameters that can be supplied with an event.
- **Event handler.** z/OSMF task or external application that can process the event parameters and display the requested information.

Figure 1 depicts the relationship of these components in the application linking process. For more information about these components and to obtain a list of the predefined event types, requestors, and handlers, see "Event types, requestors, and handlers shipped with z/OSMF" on page 7.

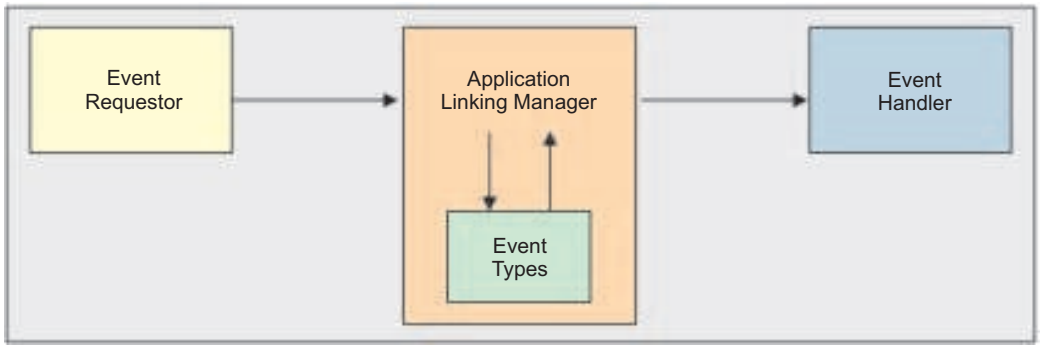

Figure 1. Key components in the application linking process

z/OSMF provides the following resources for working with the Application Linking Manager:

- **Application Linking Manager task**, which provides a graphical user interface that you can use to add, query, or remove event type and event handler definitions.
- **Application Linking Manager REST APIs**, which are a set of REST services that allows a client application to add, query, or remove event type and event handler definitions.
- **AppLinker JavaScript API**, which is a set of JavaScript services that allows a client application to send events to the Application Linking Manager or to define the context to be displayed. The JavaScript services are applicable only if you are creating your own z/OSMF plug-in.

The remainder of this section describes the Application Linking Manager REST APIs. For information about the Application Linking Manager task, see the z/OSMF online help. For details about the AppLinker JavaScript API, see "Using the Application Linking Manager JavaScript APIs" on page 700.

Operations provided through the Application Linking Manager interface services

Table 2 lists the operations that the Application Linking Manager interface services provide.

Table 2. Operations provided through the Application Linking Manager interface services

Operation	HTTP method and URI path
"Register an event type" on page 19	POST /zosmf/izual/rest/eventtype
"Register an event handler" on page 22	POST /zosmf/izual/rest/handler?eventTypeId=<*eventTypeId*>
"Obtain a list of all tasks that are eligible to be handlers" on page 25	GET /zosmf/izual/rest/adm/getHandlerEligibleTasks?eventTypeId=<*eventTypeId*>
"Obtain a list of handlers for an event type" on page 27	GET /zosmf/izual/rest/handler?eventTypeId=<*eventTypeId*>
"Unregister an event handler" on page 29	DELETE /zosmf/izual/rest/handler/<*handlerId*>?eventTypeId=<*eventTypeId*>
"Unregister an event type" on page 30	DELETE /zosmf/izual/rest/eventtype/<*eventTypeId*>

Required authorizations

To submit requests through the Application Linking Manager interface services, your user ID requires authorization to the Application Linking Manager task. Ensure that your user ID has READ access to the following resource profile in the ZMFAPLA class: `<SAF-prefix>.ZOSMF.ADMINTASKS.APPLINKING`. By default, any user ID with z/OSMF administrator authority can access the Application Linking Manager interface services.

Further, if you plan to use the Application Linking Manager interface services to list the registered event handlers for an event type, your user ID requires authorization to the z/OSMF SAF profile prefix on the target z/OS system, as follows:

- READ access to <*SAF_PREFIX*> in the APPL class.
- READ access to the <*SAF_PREFIX*>`.izuUsers` profile in the EJBROLE class.

By default, the z/OSMF SAF profile prefix is IZUDFLT.

For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP request and response data

The JSON content type (`"Content-Type: application/json"`) is used for sent data and returned data; for the detailed format of each returned object, see the JSON object description for each operation.

Error handling

For errors that occur during the processing of a request, the Application Linking Manager interface returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. For example, `HTTP/1.1 400 Bad Request` or `HTTP/1.1 500 Internal Server Error`

In addition, some errors might also include a returned JSON object that contains a message that describes the error. You can use this information to diagnose the problem or provide it to IBM Support, if required.

The following HTTP status codes are valid:

HTTP 200 OK
Success.

HTTP 400 Bad Request
Request contained incorrect parameters.

HTTP 401 Unauthorized
Submitter of the request did not authenticate to z/OSMF or is not authorized to the Application Linking Manager task.

HTTP 404 Bad URL
Target of the request (a URL) was not found.

HTTP 500 Internal server error
Programming error.

Error logging

Errors from the Application Linking Manager interface services are logged in the z/OSMF runtime log files or the z/OSMF server logs directory. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Invoking a z/OSMF application link externally

It is possible for an external application (a web application, for example, or a desktop application), to launch a z/OSMF event on a z/OS system. Here, the application issues a command comprised of the URL for the local instance of z/OSMF, combined with the appropriate event information.

The following is an example of such a request. In the example, *eventType-1* identifies the event type and *value-1* specifies the parameter input for the event handler.

```
https://...:9443/zosmf/?izual.eventType={eventType-1}&parm1={value-1}
```

This command launches z/OSMF in a new browser window on the issuer's system, and sends the event to z/OSMF after session startup. If the user is authenticated to z/OSMF, the handler is launched in the user's session. Otherwise, the user is prompted to authenticate before the handler can be launched.

Event types, requestors, and handlers shipped with z/OSMF

IBM ships several event types, requestors, and handlers with z/OSMF so that you can quickly start exploiting the application linking capability and easily navigate between multiple z/OSMF tasks.

For more details about these application linking objects, see the following sections:

Event types

An *event type* is the intermediary that connects an event requestor to an event handler. Event requestors and event handlers do not interact directly; instead, z/OSMF uses the event type to pass information, the event type and parameters, from the requestor to the handler.

Event handlers can process all, none, or a subset of the parameters that are provided with an event type. When creating an event requestor for an event type, ensure that the event requestor supplies the parameters that are required by the event handlers. To obtain a list of the parameters that are required for the IBM-supplied handlers to process an event type, see "Event handlers" on page 14.

For a list of the event types that are shipped with z/OSMF, see Table 3 on page 9. This table provides the following information for each event type: ID, display name, description, parameters, and the name of the plug-in that registers the event type. By default, these event types are listed in the Application Linking Manager task if the corresponding plug-in is configured in your z/OSMF instance.

Note: This table is formatted in landscape view to improve usability when you print copies of these pages. To adjust the view in Adobe Reader, select **View** > **Rotate View** > **Clockwise**.

Table 3. Event types shipped with z/OSMF

Event Type ID	Display Name	Description	Parameters Provided	Registered By
IBM.ZOSMF.WORKFLOWS.CREATE_WORKFLOW	Create a workflow	Perform a guided set of steps, for example, to configure components or products in your installation.	**workflow_definition_file_name** Fully-qualified UNIX file name or PDS name of the workflow definition file. **workflow_name** Name that the user has provided for the workflow. **workflow_owner** z/OS user id of the workflow owner. **workflow_target_system** Name of the system for this workflow.	Workflows plug-in
IBM.ZOSMF.CONFIGURE_NETWORK_POLICIES	Configure network policies	Configure z/OS Communications Server network policies.	No parameters are provided for this event type.	Configuration Assistant plug-in
IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE_DEFINITION	View Active Service Definition	View the active service definition.	**sysplex** Name of the sysplex for which to display the active service definition. **timestamp** Timestamp in milliseconds when the service definition was active.	Workload Management plug-in
IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE_DEFINITION.REPORT_CLASS	View Report Class of Active Service Definition	View the report classes that are contained in the active service definition.	**sysplex** Name of the sysplex for which to display the active service definition. **timestamp** Timestamp in milliseconds when the service definition was active. **reportClass** Name of the report class to be viewed.	Workload Management plug-in
IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE_DEFINITION.SERVICE_CLASS	View Service Class of Active Service Definition	View the service classes that are contained in the active service definition.	**sysplex** Name of the sysplex for which to display the active service definition. **timestamp** Timestamp in milliseconds when the service definition was active. **serviceClass** Name of the service class to be viewed. **period** Period in the service class to be viewed.	Workload Management plug-in
IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE_DEFINITION.WORKLOAD	View Workload of Active Service Definition	View the workloads that are contained in the active service definition.	**sysplex** Name of the sysplex for which to display the active service definition. **timestamp** Timestamp in milliseconds when the service definition was active. **workload** Name of the workload to be viewed.	Workload Management plug-in

Table 3. Event types shipped with z/OSMF (continued)

Event Type ID	Display Name	Description	Parameters Provided	Registered By
IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE _POLICY	View Active Service Policy	View the active service policy.	**sysplex** Name of the sysplex for which to display the active service policy. **timestamp** Timestamp in milliseconds when the service policy was active.	Workload Management plug-in
IBM.ZOSMF.VIEW_DATASET	View Data Set	View or browse a data set.	**dataSetName** Name of the dataset to be viewed.	ISPF plug-in
IBM.ZOSMF.VIEW_JOB_STATUS	View Job Status	View the status of a job.	**jobName** Name of the job for which to display status. **jobId** ID of the job for which to display status.	ISPF plug-in
IBM.ZOSMF.VIEW_SYSPLEX_PERF	View Sysplex Performance	View the overall performance of a sysplex.	**sysplex** Name of the sysplex for which to display the performance.	Resource Monitoring plug-in
IBM.ZOSMF.VIEW_SYSPLEX_PERF_INDEX	View Performance Index Details	View the performance index of a sysplex.	**sysplex** Name of the sysplex for which to display the performance index.	Resource Monitoring plug-in
IBM.ZOSMF.VIEW_WLM_REPORT_CLASS_PERF	View Report Class Performance	View the execution velocity and response time metrics for the report classes in a sysplex.	**sysplex** Name of the sysplex for which to display performance metrics. **reportClass** Name of the report class for which to display metrics.	Resource Monitoring plug-in
IBM.ZOSMF.VIEW_WLM_SERVICE_CLASS_PERF	View Service Class Performance	View the performance of active service class periods in a sysplex.	**sysplex** Name of the sysplex for which to display performance metrics. **serviceClass** Name of the service class for which to display metrics. **period** Period in the service class for which to display metrics.	Resource Monitoring plug-in
IBM.ZOSMF.VIEW_WLM_STATUS	View WLM Status	Display the status of WLM in the sysplex.	**sysplex** Name of the sysplex for which to display the status of WLM.	Workload Management plug-in
IBM.ZOSMF.VIEW_WLM_WORKLOAD_PERF	View Workload Performance	View the execution velocity and response time metrics for the workloads in a sysplex.	**sysplex** Name of the sysplex for which to display performance metrics. **workload** Name of the workload for which to display metrics.	Resource Monitoring plug-in

Event requestors

An *event requestor* provides a user-interface control that when invoked passes an event type and the required parameters to the Application Linking Manager task so that they can be passed to an appropriate handler. Table 4 on page 12 lists the name and description of the event requestors that are shipped with z/OSMF, and it provides a list of the event types the requestor can invoke, the user-interface (UI) control you can use to invoke the event type, and the IBM-supplied handlers for the event type.

When creating an event handler for the IBM-supplied event types, you can use the information provided in Table 4 on page 12 to invoke the event type and verify that your handler is displaying the expected output.

Notes:

1. The UI controls listed in Table 4 on page 12 are available only if the corresponding event type is registered in z/OSMF, and one or more handlers are registered for the event type and are available to process the request. Otherwise, the UI control is disabled or is not provided in the user interface.
2. Table 4 on page 12 is formatted in landscape view to improve usability when you print copies of these pages. To adjust the view in Adobe Reader, select **View** > **Rotate View** > **Clockwise**.

Table 4. Event requestors shipped with z/OSMF

Event Requestor	Description	Invoked Event Type	UI Controls that Invoke the Event Type	Event Handler
Application Linking Manager task	Create context-sensitive launch points between z/OSMF tasks or external applications.	IBM.ZOSMF.IMPORT_EXTERNAL _APP	• **Import** action provided in the **Event Types** table. • **Import** action provided in the **Handlers** table.	Import Manager task
Incident Log task	Diagnose system problems, and send diagnostic data to IBM or other vendors for further diagnostics.	IBM.ZOSMF.VIEW_DATASET	• **View Log** button on the Diagnostic Data tab on the View Diagnostic Details page. • Link in the **Source Name** column in the **Diagnostic Data** table on the Diagnostic Data tab.	ISPF task
Incident Log task	Diagnose system problems, and send diagnostic data to IBM or other vendors for further diagnostics.	IBM.ZOSMF.VIEW_JOB_STATUS	**View Job Details** action provided in the **FTP Job Status** table on the FTP Job Status page.	By default, no handler is provided for this event type.
Links task	Add links to external sites for system management tools and information.	IBM.ZOSMF.IMPORT_EXTERNAL _APP	**New** action provided in the Links page.	Import Manager task
Resource Monitoring task	Monitor the performance of the z/OS, AIX®, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE _DEFINITION.SERVICE_CLASS	**View WLM Service Class** action provided in the bar chart for metrics that are organized or filtered by WLM service class or WLM service class period.	Workload Management task
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE _DEFINITION.WORKLOAD	**View WLM Workload** action provided in the bar chart for metrics that are organized or filtered by WLM workload.	Workload Management task
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE _DEFINITION.REPORT_CLASS	**View WLM Report Class** action provided in the bar chart for metrics that are organized or filtered by WLM report class or WLM report class period.	Workload Management task
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_SYSPLEX_PERF	**System Status** link provided in the Select Sysplex window.	System Status task
System Status task	Quickly assess the workload performance on the systems in your enterprise, and define the systems to be monitored.	IBM.ZOSMF.VIEW_SYSPLEX_PERF_INDEX	• **View** > **Performance Index Details** action provided in the **Resources** table on the System Status page. • Link in the **Performance Index Status** column in the **Resources** table on the System Status page.	Resource Monitoring task

Table 4. Event requestors shipped with z/OSMF (continued)

Event Requestor	Description	Invoked Event Type	UI Controls that Invoke the Event Type	Event Handler
System Status task	Quickly assess the workload performance on the systems in your enterprise, and define the systems to be monitored.	IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE_DEFINITION	• **View** > **Active WLM Service Definition** action provided in the **Resources** table on the System Status page. • Link in the **Related Service Definition** column in the **Resources** table on the System Status page.	Workload Management task
System Status task	Quickly assess the workload performance on the systems in your enterprise, and define the systems to be monitored.	IBM.ZOSMF.VIEW_ACTIVE_WLM_SERVICE_POLICY	• **View** > **Active WLM Policy** action provided in the **Resources** table on the System Status page. • Link in the **Active WLM Policy** column in the **Resources** table on the System Status page.	Workload Management task
System Status task	Quickly assess the workload performance on the systems in your enterprise, and define the systems to be monitored.	IBM.ZOSMF.VIEW_WLM_STATUS	**View** > **WLM Status** action provided in the **Resources** table on the System Status page.	Workload Management task
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_SYSPLEX_PERF	**View performance of systems** link provided on the WLM Status page.	System Status task
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_WLM_SERVICE_CLASS_PERF	• **View performance of active policy** link provided on the WLM Status page. • **View Performance of Active Policy** action provided in the table on the Service Policies for Sysplex page. • **View Performance of Active Policy** action provided in the Service Policies table in a View or Modify tab. • **View Performance of Selected** and **View Performance of All** actions provided in the Service Classes and Service Class Overrides tables in a View or Modify tab.	Resource Monitoring task
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_WLM_WORKLOAD_PERF	**View Performance of Selected** and **View Performance of All** actions provided in the Workloads table in a View or Modify tab.	Resource Monitoring task
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_WLM_REPORT_CLASS_PERF	**View Performance of Selected** and **View Performance of All** actions provided in the Report Classes table in a View or Modify tab.	Resource Monitoring task

Event handlers

An *event handler* is a z/OSMF task or external application that can handle requests sent by event requestors. Event handlers support specific event types and can support all, none, or a subset of the parameters that are supplied with an event type. For a handler to process a request, it must receive the correct parameters from the event requestor.

For a list of the event types and parameters (optional and required) that are supported by the handlers shipped with z/OSMF, see Table 5 on page 15. This table also provides a description of each handler and the expected output for each event type and handler combination.

Note: This table is formatted in landscape view to improve usability when you print copies of these pages. To adjust the view in Adobe Reader, select **View** > **Rotate View** > **Clockwise**.

Table 5. Event handlers shipped with z/OSMF

Event Handler	Description	Supported Event Type	Required Parameters	Optional Parameters	Expected Output
Configuration Assistant task	Configure TCP/IP policy-based networking functions.	IBM.ZOSMF.CONFIGURE_NETWORK_POLICIES	None	None	Displays the main page in the Configuration Assistant task.
Import Manager task	Add installation-specific function to z/OSMF in the form of plug-ins.	IBM.ZOSMF.IMPORT_EXTERNAL_APP	tab	None	Opens to the Import Manager task.
ISPF task	Access traditional ISPF applications.	IBM.ZOSMF.VIEW_DATASET	dataSetName	None	Displays the data set.
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_SYSPLEX_PERF_INDEX	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex.	Opens a dashboard that contains the performance index for the following items: • Important service class periods • All service class periods • Report class periods
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_WLM_REPORT_CLASS_PERF	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **reportClass** If unspecified, the metrics for all report classes are displayed.	Opens a dashboard that contains the following metrics for each WLM report class: • Execution velocity • Response time
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_WLM_SERVICE_CLASS_PERF	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **serviceClass** If unspecified, the metrics for all service classes are displayed. **period** If unspecified, the metrics for all periods in the specified service class are displayed. This value is used only if a service class is specified.	Opens a dashboard that contains the following metrics for each WLM service class period: • Performance index • Execution velocity • Execution velocity goal • Response time • Response time goal

Table 5. Event handlers shipped with z/OSMF (continued)

Event Handler	Description	Supported Event Type	Required Parameters	Optional Parameters	Expected Output
Resource Monitoring task	Monitor the performance of the z/OS, AIX, Linux, and Windows systems in your enterprise.	IBM.ZOSMF.VIEW_WLM _WORKLOAD_PERF	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **workload** If unspecified, the metrics for all workloads are displayed.	Opens a dashboard that contains the following metrics for each WLM workload: • Execution velocity • Response time
System Status task	Quickly assess the workload performance on the systems in your enterprise, and define the systems to be monitored.	IBM.ZOSMF.VIEW_SYSPLEX_PERF	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex.	Displays a list of the sysplexes that are defined in the System Status task.
Workflows task	Perform a guided set of steps, for example, to configure components or products in your installation.	IBM.ZOSMF.WORKFLOWS. CREATE_WORKFLOW	**workflow_definition _file_name** Fully-qualified UNIX file name or PDS name of the workflow definition file. **workflow_name** Name that the user has provided for the workflow. **workflow _owner** z/OS user id of the workflow owner. **workflow _target_system** Name of the system for this workflow.	**workflow _callback** Callback URL to use when the workflow is created. For example, the caller can be notified with the name of workflow, to be used for a subsequent invocation to launch the workflow. **workflow _comments** Comments to be added to the workflow on creation.	Opens to the *Workflow Details* panel for the newly created workflow in the Workflows task.
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_ACTIVE_WLM _SERVICE_DEFINITION	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **timestamp** If unspecified, the service definition that is currently active is displayed.	Displays the service definition that is active in the sysplex.

Table 5. Event handlers shipped with z/OSMF (continued)

Event Handler	Description	Supported Event Type	Required Parameters	Optional Parameters	Expected Output
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_ACTIVE_WLM _SERVICE_DEFINITION.REPORT _CLASS	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **timestamp** If unspecified, the service definition that is currently active is displayed. **reportClass** If unspecified, all the report classes in the service definition are displayed.	Displays the specified report class or a list of all the report classes defined in the active service definition.
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_ACTIVE_WLM _SERVICE_DEFINITION.SERVICE _CLASS	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **timestamp** If unspecified, the service definition that is currently active is displayed. **serviceClass** If unspecified, all the service classes in the service definition are displayed. **period** If unspecified, all the periods in the service class are displayed. This value is used only if a service class is specified.	Displays the specified service class and period, or a list of all the service classes defined in the active service definition.
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_ACTIVE_WLM _SERVICE_DEFINITION.WORKLOAD	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **timestamp** If unspecified, the service definition that is currently active is displayed. **workload** If unspecified, all the workloads in the service definition are displayed.	Displays the specified workload or a list of all the workloads defined in the active service definition.
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_ACTIVE_WLM _SERVICE_POLICY	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex. **timestamp** If unspecified, the service policy that is currently active is displayed.	Displays the service policy that is active in the sysplex.

Table 5. Event handlers shipped with z/OSMF (continued)

Event Handler	Description	Supported Event Type	Required Parameters	Optional Parameters	Expected Output
Workload Management task	Administer and operate WLM, and manage WLM service definitions and policies.	IBM.ZOSMF.VIEW_WLM_STATUS	None	**sysplex** If unspecified, the default value is the z/OSMF host sysplex.	Opens the WLM Status tab.

Register an event type

You can use this operation to define a new event type to z/OSMF.

HTTP method and URI path

```
POST /zosmf/izual/rest/eventtype
```

where:

- **/zosmf/izual/rest** identifies the Application Linking Manager interface.
- **eventtype** identifies the event type component of the application linking process.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that describes the event type to be registered, for example:

```
{
id: "IBM.ZOSMF.EVENT_TYPE_ID",
displayName: "Default English name",
desc: "Default English description",
owner: "ownerId",
params: {
  "key1": "English description of the parameter.",
  "key2": "English description of the parameter."}
}
```

Figure 2. Registering an event type: request content

The following values are supported:

id Specify a unique identifier for the event type. It can contain up to 50 characters, including alphanumeric characters (A-Z, a-z, and 0-9), periods (.), and underscores (_). The event ID is required and must be unique.

It is recommended that IDs have the format *company-name.product-name.event-name* where:

- *company-name* is the name of your company. Use a period as the delimiter within the company name. For example, for Tivoli products, the company name can be *IBM.TIVOLI*.
- *product-name* is the name of the product for which the event type is being created.
- *event-name* is the action that will be completed by the event handler. The event name should start with a verb that reflects this action. Use an underscore as the delimiter within the event name.

  For example, to create an event type that allows a user to view the status of a job listed in z/OSMF, the event name portion of the ID can be *VIEW_JOB_STATUS*. The entire ID can be *IBM.ZOSMF.VIEW_JOB_STATUS*.

displayName
Specify the name of the event type. The name is required and can contain up to 50 characters.

Example: *View job status*.

desc
Specify a description of the event type. The description is optional and can contain up to 200 characters.

Example: *Use this event type to view the status of a job. This event type is invoked when a user selects the View Job Status action.*

owner
Specify the ID of the first z/OSMF task or external application that registered the event type. Typically, event types are registered by or on behalf of event handlers. They can also be registered by or on behalf of event requestors. This field is required, and can contain up to 50 characters, including alphanumeric characters (A-Z, a-z, and 0-9), periods (.), and underscores (_).

params
Specify the name and description of each parameter that event requestors can supply with an event. Enter each parameter name and description combination on a separate line, and enclose the entry in quotes. Use a colon to separate the parameter name and description, and a comma to end each entry. The final entry is not ended with a delimiter.

This area can contain up to 4,000 characters, including alphanumeric characters (A-Z, a-z, and 0-9), periods (.), underscores (_), and commas (,).

For example, to allow event requestors to provide a job ID, job name, and user ID for an event type that displays the status of a job, you would specify the following parameters:

```
params: {
"jobID": "ID assigned to the job.",
"jobName": "Name specified for the job.",
"userID": "ID of the user who submitted the job."
}
```

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 6.

IBM-supplied event types

z/OSMF includes a number of predefined event types, requestors, and handlers. For a list, see *IBM z/OS Management Facility Configuration Guide*.

Example of registering an event type

A sample request to register an event type is shown in Figure 3 on page 21.

```
POST /zosmf/izual/rest/eventtype HTTP/1.1
Host: zosmf1.yourco.com

Accept: application/json
Content-Type: application/json

{
 "id":"IBM.ZOSMF.VIEW_JOB_STATUS",
 "displayName":"View Job Status",
 "owner":"SDSF",
 "params":{"jobName": "Name of the job for which to view status."}
}
```

Figure 3. Example: Registering an event type

Expected response

On completion, the Application Linking Manager interface returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. See "Error handling" on page 6.

The response also includes a JSON object with additional information about the results of the request. If your request is successful, the JSON object contains null data for the error and result fields, as shown in Figure 4.

```
{"error":null,"result":null}
```

Figure 4. Example: Returned results of a successful event registration

For an unsuccessful request, the JSON object contains an error message in the error fields, `msgid` and `msgtext`. The example in Figure 5 shows the results for an attempt to register an already-registered event with different parameters:

```
{
"error":{
  "msgid":"IZUG690E",
  "msgtxt":"Event type \"IBM.ZOSMF.VIEW_JOB_STATUS\" is already defined, but
    different parameters are specified."},
"result":null
}
```

Figure 5. Example: Returned results of an unsuccessful event registration

Register an event handler

You can use this operation to define a new event handler to z/OSMF.

HTTP method and URI path

```
POST /zosmf/izual/rest/handler?eventTypeId=<eventTypeId>
```

---

where:

- **/zosmf/izual/rest** identifies the Application Linking Manager interface.
- **handler** identifies the event handler component of the application linking process.
- **eventTypeId=<eventTypeId>** is the event type to be associated with the new handler.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that describes the event handler to be registered.

The following values are supported:

type
Handler type. For a z/OSMF plug-in, specify `INTERNAL`. For an external application, specify `EXTERNAL`.

id Unique identifier for a launch point within the handler task or application. It can contain up to 50 characters, including alphanumeric characters (A-Z, a-z, and 0-9), periods (.), and underscores (_). The handler ID is required and must be unique.

For external applications, you can specify any value. To ensure uniqueness, it is recommended that you make the company name and the application name part of the handler ID. For example, *IBM.TIVOLI.OMEGAMON*.

For applications within the ISPF task, it is recommended that you prefix the handler ID with IBM.ISPF. Then, specify an ID for the application that will handle the events. For example, *IBM.ISPF.SDSF.ST*.

applId
Identifier assigned to the z/OSMF plug-in that contains the task. It is required for a z/OSMF task (type is set to `INTERNAL`). Omit this value if the handler is an external application.

Table 6 lists the valid applId values for the z/OSMF tasks.

Table 6. Valid applId values for the z/OSMF plug-ins

z/OSMF Task	applId Value
Configuration Assistant task	CAV1R11
Import Manager task	IzuImportManager
ISPF task	com.ibm.zoszmf.ispf

Table 6. Valid applId values for the z/OSMF plug-ins (continued)

z/OSMF Task	applId Value
Resource Monitoring task	IZUR
System Status task	IZUR
Workflows task	workflow
Workload Management task	IZUW

taskId

Identifier assigned to the z/OSMF task. It can contain up to 50 characters, including alphanumeric characters (A-Z, a-z, and 0-9), periods (.), and underscores (_). The task ID is required when type is set to `INTERNAL`. Omit this value if the handler is an external application.

Table 7 lists the valid task ID values for the z/OSMF tasks.

Table 7. Valid taskId values for the z/OSMF tasks

z/OSMF Task	taskId Value
Configuration Assistant task	Configuration Assistant
Import Manager task	IZUG_TASK_zOSMFImportManager
ISPF task	ISPF
Resource Monitoring task	IZUR_PERFDESKS_TASK_ID
System Status task	IZUR_OVERVIEW_TASK_ID
Workflows task	Workflows
Workload Management task	Workload Management

displayName

For the handler name, specify the name of the handler task or application. The name is required and can contain up to 50 characters. For z/OSMF tasks, it is recommended that you use the same name displayed in thez/OSMF navigation area. For example, *Workload Management*.

For external applications, it is recommended that you use the name of the product or application. For example, *Omegamon*.

url

URL to be used for accessing the handler. The URL can contain up to 4,000 characters, including alphanumeric characters (A-Z, a-z, 0-9), blanks, mathematical symbols (+ - = | ~ ( ) { } \), punctuation marks (? , . ! ; : ' " / [ ]), and the following special characters: %, $, #, @, ^, *, and _. The URL is required and must be URI-encoded as specified in RFC 2396. For more information about RFC 2396, see the Uniform Resource Identifiers (URI): Generic Syntax web page.

For a z/OSMF task, specify a URL that is relative to the z/OSMF instance. That is, the URL must begin with /zosmf/. For an external application, specify the full URL, including the protocol.

options

The CONTEXT_SUPPORT option indicates what the handler will display when it processes events of this type. Specify one of the following values for CONTEXT_SUPPORT:

OPT_CONTEXT_SUPPORT_NONE

Handler is launched without context. That is, its homepage is displayed. If the handler is already open, it receives focus, but the context is not updated.

If the handler is an external application, it opens in a separate window. Otherwise, the handler opens in a new z/OSMF task tab.

This option is selected by default.

OPT_CONTEXT_SUPPORT_LAUNCH
Handler is launched with context. If the handler is already open, it receives focus, but the context is not updated.

If the handler is an external application, it opens in a separate window. Otherwise, the handler opens in a new z/OSMF task tab.

OPT_CONTEXT_SUPPORT_LAUNCH_AND_RELOAD
Handler is launched with context. If the handler is already open, a message is displayed warning the user that the current context will be overwritten. This option is supported only when the event requestor and handler are z/OSMF tasks.

OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH
Handler is launched with the context it specified when subscribing to the event type. This option is supported only when the event requestor and handler are z/OSMF tasks.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 6.

Expected response

On completion, the Application Linking Manager interface returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. See "Error handling" on page 6.

IBM-supplied event handler registrations

z/OSMF includes a number of predefined event types, requestors, and handlers. For a list, see "Event types, requestors, and handlers shipped with z/OSMF" on page 7.

Obtain a list of all tasks that are eligible to be handlers

The *handlerEligible* property indicates whether a z/OSMF task can participate in the application linking process as an event handler. To obtain a list of the tasks with the *handlerEligible* property set to *true*, use the GET method.

HTTP method and URI path

```
GET /zosmf/izual/rest/adm/getHandlerEligibleTasks?eventTypeId=<eventTypeId>
```

---

where:

- **/zosmf/izual/rest/adm** identifies the Application Linking Manager interface.
- **getHandlerEligibleTasks** indicates that the service will retrieve a list of tasks that are eligible to be event handlers.
- **eventTypeId=<eventTypeId>** is the event type for which the request is being submitted.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 6.

Expected response

On completion, the Application Linking Manager interface returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. See “Error handling” on page 6.

The response also includes a JSON object with additional information about the results of the request. If your request is successful, the JSON object contains null data for the error field, and the result field lists the task ID, navigation URL, display name, and plug-in ID for each task that is eligible to be a handler. Figure 6 on page 26 provides a sample response for a successful request.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"error":null,"result":{"Task":[
{"TaskID":"test2","navigationUrl":"\/zosmf\/test2","displayName":"test2","PluginID":"TestPlugin1"},
{"TaskID":"test3","navigationUrl":"\/zosmf\/test3","displayName":"test3","PluginID":"TestPlugin1"},
{"TaskID":"test4","navigationUrl":"\/zosmf\/test4","displayName":"test4","PluginID":"TestPlugin1"},
{"TaskID":"test1","navigationUrl":"\/zosmf\/test1","displayName":"Test1","PluginID":"TestPlugin2"}
]}}
```

Figure 6. Sample response from a successful list tasks request

For an unsuccessful request, the JSON object contains an error message in the error fields, `msgid` and `msgtext`.

Obtain a list of handlers for an event type

You can use this operation to obtain a list of registered handlers for an event type.

HTTP method and URI path

```
GET /zosmf/izual/rest/handler?eventTypeId=<eventTypeId>
```

---

where:

- **/zosmf/izual/rest** identifies the Application Linking Manager interface.
- **handler** identifies the event handler component of the application linking process.
- **eventTypeId=<eventTypeId>** is the ID of the event type for which you want to obtain a list of registered handlers.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 6.

Expected response

On completion, the Application Linking Manager interface returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. See "Error handling" on page 6.

The response also includes a JSON object with additional information about the results of the request. If your request is successful, the JSON object contains null data for the error field, as shown in Figure 7 on page 28, Figure 8 on page 28, and Figure 9 on page 28.

If the request is successful and one or more handlers are enabled for the event type, information about the registered handlers are returned, as depicted in Figure 7 on page 28.

```
{"result":[{"id":"IBM.ZOSMF.IZU_IMPORT_HANDLER","taskId":"IZUG_TASK_zOSMFImportManager",
"enabled":true,"defaultHandler":false,"applId":"IzuImportManager","type":"INTERNAL",
"displayName":"Import Manager","url":"\/zosmf\/IzuImportUtility\/index.jsp",
"eventTypeId":"IBM.ZOSMF.IMPORT_EXTERNAL_APP",
"options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH"}}],"error":null}
```

Figure 7. Example: Handlers enabled for the event type

If the request is successful and no handlers are registered for the event type, the result field contains null data, as depicted in Figure 8.

```
{"result":null,"error":null}
```

Figure 8. Example: Returned results of a successful list handlers request

If the request is successful and all the handlers that are registered for the event type are disabled, the result field contains an empty array, as depicted in Figure 9.

```
//Result if all the handlers defined for the event type are disabled.
{"result":[], "error":null}
```

Figure 9. Example: Returned results of a successful list handlers request

For an unsuccessful request, the JSON object contains an error message in the error fields, `msgid` and `msgtext`.

Unregister an event handler

You can use this operation to remove an existing event handler registration from z/OSMF.

HTTP method and URI path

```
DELETE /zosmf/izual/rest/handler/<handlerId>?eventTypeId=<eventTypeId>
```

---

where:

- **/zosmf/izual/rest** identifies the Application Linking Manager interface.
- **handler** identifies the event handler component of the application linking process.
- **<handlerId>** is the ID of the event handler to be removed.
- **eventTypeId=<eventTypeId>** is the ID of the event type for which you want to remove the specified handler. The combination of **<handlerId>** and **<eventTypeId>** identifies the handler registration to be removed.

Standard headers

None.

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 6.

Expected response

On completion, the Application Linking Manager interface returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. See "Error handling" on page 6.

Unregister an event type

You can use this operation to remove the definition of an event type from z/OSMF.

HTTP method and URI path

```
DELETE /zosmf/izual/rest/eventtype/<eventTypeId>
```

---

where:

- **/zosmf/izual/rest** identifies the Application Linking Manager interface.
- **eventtype** identifies the event type component of the application linking process.
- **<eventTypeId>** is the ID of the event type to be removed.

Standard headers

None.

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 6.

Expected response

On completion, the Application Linking Manager interface returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. See "Error handling" on page 6.

Application server routing services

The application server routing services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. Use these services to route requests and responses between the client-side and server-side code for any z/OSMF plug-ins you created where the server-side code is hosted on an application server other than the z/OSMF server.

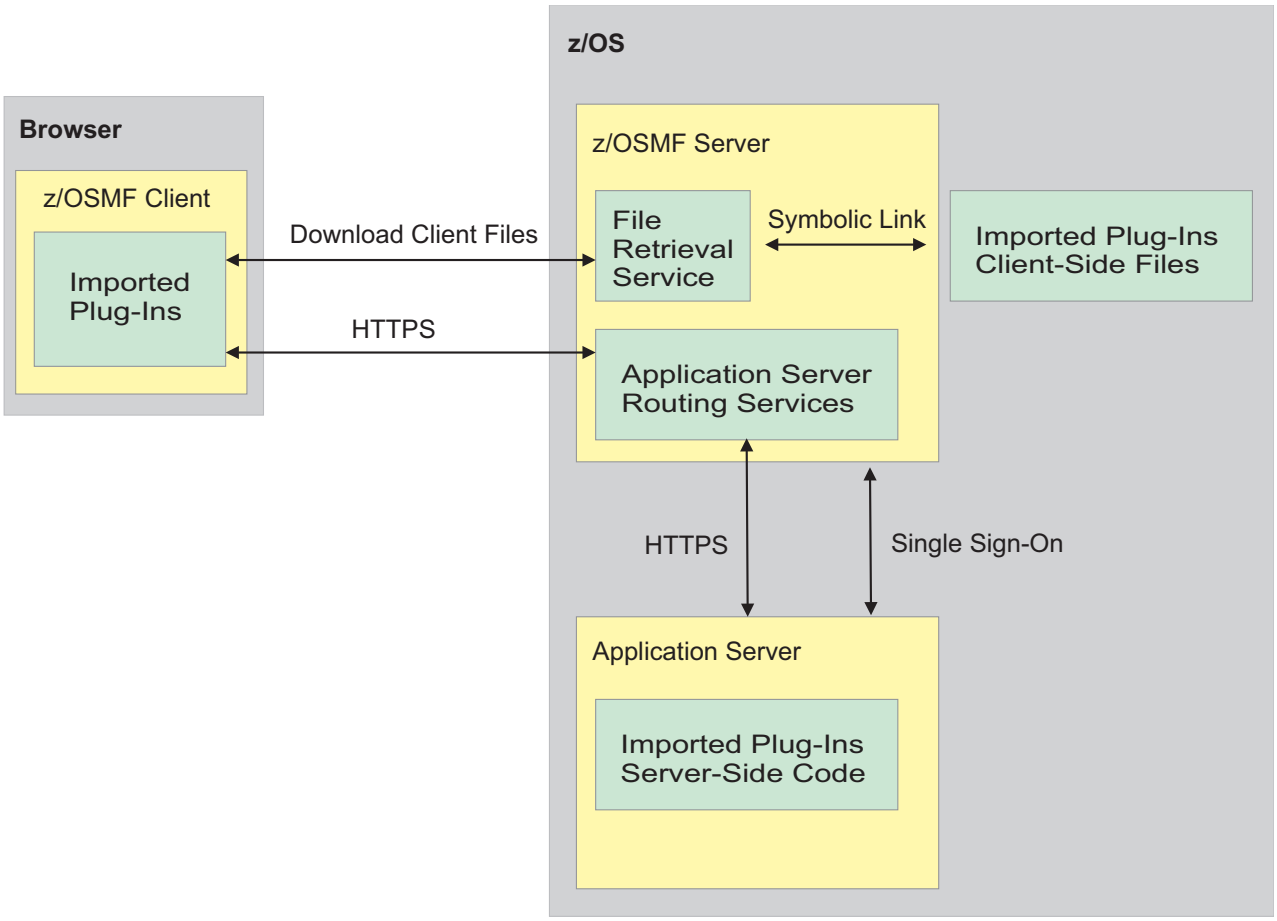

Figure 10. Process for routing requests and responses between application servers

As depicted in Figure 10, the process of routing requests and responses for plug-ins where the client and server-side code are on different application servers is as follows:

1. Use the z/OSMF Import Manager task to import your plug-in into z/OSMF, and to associate each task in the plug-in with an application server. For instructions, see "Adding your applications to z/OSMF" on page 735.
2. During the import process, z/OSMF core creates symbolic links to the client-side code for your application and stores those links in the z/OSMF file system on the z/OSMF server.
3. When the import process completes, z/OSMF core adds the tasks included in your plug-in to the z/OSMF navigation area.
4. When a user selects your task in the z/OSMF navigation area, z/OSMF core submits an HTTPS request to the file retrieval service to retrieve the client-side files and the browser downloads those files.

5. When the user performs an action that requires the task to interact with the server-side code, the client-side code for the task submits an HTTPS request to the application server routing interface, and that interface routes the request to the application server that is associated with the task.
6. The server-side code for your task processes the request and submits an HTTPS response to the application server routing interface, and that interface routes the response to the client.

The z/OSMF server and the application server that hosts the server-side code for your plug-in (referred to as the *target application server*) can reside on the same system or on different systems. To enable single sign-on between the servers, ensure that the servers share the same Lightweight Third Party Authentication (LTPA) key files.

Operations provided through the application server routing services

Table 8 lists the operations that the application server routing services provide.

Table 8. Operations provided through the application server routing services.

Operation	HTTP method and URI path
"Retrieve data from an application server" on page 35	GET /zosmf/externalgateway/system?content=<*http-content*>
"Update data for an application server" on page 40	POST /zosmf/externalgateway/system PUT /zosmf/externalgateway/system
"Delete data from an application server" on page 43	DELETE /zosmf/externalgateway/system?content=<*http-content*>

Required authorizations

The user must be logged into z/OSMF. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP response data

The JSON content type (`"Content-Type: application/json"`) is used for HTTP response data. If the client requests for the application server routing service to add additional information to the response from the target application server, the service wraps the response in the following JSON object. Otherwise, the service returns only the response from the target application server.

```
{
 "primaryAPIVersion":"primary-API-version",
 "systemsOutput":
   {
     "systemOutput":"system-output",
     "rc":"return-code",
     "error":{"msgid":"message-ID","msgtxt":"message-text"},
     "systemName":"system-name"
   },
 "numOfSystems":"total-systems"
}
```

where

primary-API-version
: Version of the application server routing services interface on the z/OSMF server.

systemsOutput
Contains a set of attributes that provide different information about the response from the target application server.

system-output
Contains the response from the target application server.

return-code
Contains the return code provided by the target application server. The return code can be one of the following values:
OK Success.
HttpConnectionFailed
The HTTPS connection failed. Typically, this error occurs when the target application server is unavailable or a network error has occurred.
HttpConnectionTimedOut
The HTTPS request did not complete in the time allotted.
CertificateError
The certificate for the target application server is not trusted.
InvalidLogin
The login credentials for the target application server are not valid.
FailedWithMessage
The request was successful; however, an internal error occurred on the target application server.
UnexpectedFailure
An unexpected error occurred.

error If an error occurred with the request, the error attribute contains the message ID (msgid) and message text (msgtxt) for the message that was issued. Otherwise, this attribute is *null*.

system-name
Nickname assigned to the system entry in the z/OSMF Systems task that describes the settings required to access the target application server.

total-systems
Value is set to *1* because the HTTPS request can be sent to only one application server at a time.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains a message that describes the error.

The following HTTP status codes are valid:
HTTP 200 OK
Success.
HTTP 400 Bad request
Request contained incorrect parameters.
HTTP 401 Unauthorized
Submitter of the request is not authorized to use the service or did not authenticate with z/OSMF, or single sign-on is not enabled between the z/OSMF server and the target application server.
HTTP 404 Bad URL
Target of the request (a URL) was not found.
HTTP 500 Internal server error
Programming error.

Error logging

Errors from the application server routing services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Retrieve data from an application server

You can use this operation to request that z/OSMF route a retrieve data request to the application server where the server-side code for your plug-in resides.

HTTP method and URI path

```
GET /zosmf/externalgateway/system?content=<http-content>
```

where:

- **zosmf/externalgateway** identifies the application server routing services.
- **system** informs the service that the request will be routed to only one application server.
- **content=<*http-content*>** represents the parameters used to qualify the request. Table 9 lists the parameters that are supported for this request.

  Important: If the value for a parameter contains a number sign (#), encode the number sign as *%23*. Otherwise, everything following the number sign will be omitted from the request. For example, if the target is *AppServer#1*, specify *AppServer%231*.

Table 9. Supported input parameters for the application server routing services

Parameter	Required	Description

target	Yes	Nickname assigned to the system entry in the z/OSMF Systems task that describes the settings required to access the application server where the server-side code for your plug-in resides. If the specified system entry does not exist, the request will fail.
		
		z/OSMF stores the nickname for the target application server in the window object in the Browser Object Model. To retrieve the nickname, issue the following JavaScript command from your task:
		`window.frameElement.getAttribute("target")`
		
		For example:
		`postCreate: function() {`
		`  var target = window.frameElement.getAttribute("target");`
		`  var remoteURL = "/zosmf/externalgateway/system?content=`
		`  {'target':'" + target + "','resourcePath':'/testApp'}";`
		`}`

resourcePath	Yes	Path to the service that will process the request.
requestProperties	No	HTTP headers to be included in the HTTP request. Specify the HTTP headers as name and value pairs. If HTTP headers are omitted or are *null*, default values will be used, which are valid for most installations.
timeout	No	Amount of time in milliseconds allowed to process a request. The value can range from 1 to 5601000 milliseconds. If omitted, the default value of 20000 milliseconds is used.
wrapped	No	Indicator of whether the application server routing service will wrap the response from the target application server in a JSON object that contains additional information about the response. Set the parameter to *N* to obtain only the response provided by the target application server. Otherwise, set the parameter to *Y* or omit it to obtain the response along with additional information. For more details, see "Content type used for HTTP response data" on page 32.

Table 9. Supported input parameters for the application server routing services (continued)

Parameter	Required	Description
binary	No	Indicator of whether the response from the target application server is in binary format. Set the parameter to *N* or omit it if the response is not in binary format. Otherwise, set the parameter to *Y* if the response is in binary format.
content	Yes if the HTTP method is POST or PUT.	Parameters or JSON object to include in the body of the HTTPS request that will be sent to the service that will process the request.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Application server routing services" on page 31.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 33.

The response also includes a JSON object that contains the requested information. For more details, see "Content type used for HTTP response data" on page 32.

Example 1: Retrieve wrapped data from an application server

To retrieve wrapped data from the application server identified in system entry *appServer1*, which is defined in the z/OSMF Systems task, submit the following request:

```
GET /zosmf/externalgateway/system?content={"target":"appServer1","resourcePath":"/testApp"} HTTP/1.1

Host: appname.yourco.com
```

Figure 11. Sample request to retrieve wrapped data from an application server

A sample response is shown in Figure 12 on page 37.

```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":
    {
      "systemOutput":
        {
          "items":[
            {
              "object-ID":"objectA"
              "attribute1":"value1",
              "attribute2":"value2",
              "attribute3":"value3",
              "attribute4":"value4",
              "attribute5":"value5",
            },
            {
              "object-ID":"objectB"
              "attribute1":"value6",
              "attribute2":"value7",
              "attribute3":"value8",
              "attribute4":"value9",
              "attribute5":"value10",
            }
          ],
        },
      "rc":"Ok",
      "error":null,
     "systemName":"appServer1"
    },
  "numOfSystems":1
}
```

Figure 12. Sample response for retrieving wrapped data from an application server

Example 2: Retrieve unwrapped data from an application server

To retrieve unwrapped data from the application server identified in system entry *appServer1*, which is defined in the z/OSMF Systems task, submit the following request:

```
GET /zosmf/externalgateway/system?content={"target":"appServer1","resourcePath":"/testApp",
"wrapped":"N"} HTTP/1.1

Host: appname.yourco.com
```

Figure 13. Sample request to retrieve unwrapped data from an application server

A sample response is shown in Figure 14 on page 38.

```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2015 05:39:28 +0000GMT
Connection: close


{
  "items":[
  {
    "object-ID":"objectA"
    "attribute1":"value1",
    "attribute2":"value2",
    "attribute3":"value3",
    "attribute4":"value4",
    "attribute5":"value5",
  },
  {
    "object-ID":"objectB"
    "attribute1":"value6",
    "attribute2":"value7",
    "attribute3":"value8",
    "attribute4":"value9",
    "attribute5":"value10",
  }]
}
```

Figure 14. Sample response for retrieving unwrapped data from an application server

Example 3: Retrieve binary data from an application server

To retrieve binary data from the application server identified in system entry *appServer1*, which is defined in the z/OSMF Systems task, submit the following request:

```
GET /zosmf/externalgateway/system?content={"target":"appServer1","resourcePath":"/testApp",
"binary":"Y"} HTTP/1.1

Host: appname.yourco.com
```

Figure 15. Sample request to retrieve binary data from an application server

A sample response is shown in Figure 16 on page 39.

```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2015 05:39:28 +0000GMT
Connection: close


01111011 00001101 00001010 00100000 00100000 00100010 01101001 01110100 01100101 01101101
01110011 00100010 00111010 01011011 00001101 00001010 00100000 00100000 01111011 00001101
00001010 00100000 00100000 00100000 00100000 00100010 01101111 01100010 01101010 01100101
01100011 01110100 00101101 01001001 01000100 00100010 00111010 00100010 01101111 01100010
01101010 01100101 01100011 01110100 01000001 00100010 00001101 00001010 00100000 00100000
00100000 00100000 00100010 01100001 01110100 01110100 01110010 01101001 01100010 01110101
01110100 01100101 00110001 00100010 00111010 00100010 01110110 01100001 01101100 01110101
01100101 00110001 00100010 00101100 00001101 00001010 00100000 00100000 00100000 00100000
00100010 01100001 01110100 01110100 01110010 01101001 01100010 01110101 01110100 01100101
00110010 00100010 00111010 00100010 01110110 01100001 01101100 01110101 01100101 00110010
00100010 00101100 00001101 00001010 00100000 00100000 00100000 00100010 01100001
01110100 01110100 01110010 01101001 01100010 01110101 01110100 01100101 00110011 00100010
00111010 00100010 01110110 01100001 01101100 01110101 01100101 00110011 00100010 00101100
00001101 00001010 00100000 00100000 00100000 00100000 00100010 01100001 01110100 01110100
01110010 01101001 01100010 01110101 01110100 01100101 00110100 00100010 00111010 00100010
01110110 01100001 01101100 01110101 01100101 00110100 00100010 00101100 00001101 00001010
00100000 00100000 00100000 00100000 00100010 01100001 01110100 01110100 01110010 01101001
01100010 01110101 01110100 01100101 00110101 00100010 00111010 00100010 01110110 01100001
01101100 01110101 01100101 00110101 00100010 00101100 00001101 00001010 00100000 00100000
00100000 00100000 00100010 01100001 01110100 01110100 01110010 01101001 01100010 01110101
01110100 01100101 00110101 00100010 00111010 00100010 01110110 01100001 01101100 01110101
01100101 00110001 00110000 00100010 00101100 00001101 00001010 00100000 00100000 01111101
01011101 00001101 00001010 01111101
```

Figure 16. Sample response for retrieving binary data from an application server

Update data for an application server

You can use this operation to request that z/OSMF route an update data request to the application server where the server-side code for your plug-in resides.

HTTP method and URI path

```
POST /zosmf/externalgateway/system
PUT /zosmf/externalgateway/system
```

where:

- **zosmf/externalgateway** identifies the application server routing services.
- **system** informs the service that the request will be routed to only one application server.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that describes the objects to be created or modified on the target application server. Table 10 lists the supported parameters.

Table 10. Supported input parameters for the application server routing services

Parameter	Required	Description

target	Yes	Nickname assigned to the system entry in the z/OSMF Systems task that describes the settings required to access the application server where the server-side code for your plug-in resides. If the specified system entry does not exist, the request will fail.
		
		z/OSMF stores the nickname for the target application server in the window object in the Browser Object Model. To retrieve the nickname, issue the following JavaScript command from your task:
		`window.frameElement.getAttribute("target")`
		
		For example:
		`postCreate: function() {`
		`  var target = window.frameElement.getAttribute("target");`
		`  var remoteURL = "/zosmf/externalgateway/system?content=`
		`  {'target':'" + target + "','resourcePath':'/testApp'}";`
		`}`

resourcePath	Yes	Path to the service that will process the request.
requestProperties	No	HTTP headers to be included in the HTTP request. Specify the HTTP headers as name and value pairs. If HTTP headers are omitted or are *null*, default values will be used, which are valid for most installations.
timeout	No	Amount of time in milliseconds allowed to process a request. The value can range from 1 to 5601000 milliseconds. If omitted, the default value of 20000 milliseconds is used.

Table 10. Supported input parameters for the application server routing services (continued)

Parameter	Required	Description
wrapped	No	Indicator of whether the application server routing service will wrap the response from the target application server in a JSON object that contains additional information about the response. Set the parameter to *N* to obtain only the response provided by the target application server. Otherwise, set the parameter to *Y* or omit it to obtain the response along with additional information. For more details, see "Content type used for HTTP response data" on page 32.
binary	No	Indicator of whether the response from the target application server is in binary format. Set the parameter to *N* or omit it if the response is not in binary format. Otherwise, set the parameter to *Y* if the response is in binary format.
content	Yes if the HTTP method is POST or PUT.	Parameters or JSON object to include in the body of the HTTPS request that will be sent to the service that will process the request.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Application server routing services" on page 31.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 33.

The response also includes a JSON object that contains the requested information. For more details, see "Content type used for HTTP response data" on page 32.

Example

To add *objectC* on the application server identified in system entry *appServer1*, which is defined in the z/OSMF Systems task, submit the following request:

```
POST /zosmf/externalgateway/system HTTP/1.1

Host: appname.yourco.com

{"target":"appServer1","resourcePath":"/testApp/objectC","content":{
"attribute1":"value11","attribute2":"value12","attribute3":"value13",
"attribute4":"value14","attribute5":"value15"}}
```

Figure 17. Sample request to update data on an application server

A sample response is shown in Figure 18 on page 42.

```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":
    {
      "systemOutput":
        {
          "result":"success"
        },
      "rc":"Ok",
      "error":null,
     "systemName":"appServer1"
    },
  "numOfSystems":1
}
```

Figure 18. Sample response for updating data on an application server

Delete data from an application server

You can use this operation to request that z/OSMF route a delete data request to the application server where the server-side code for your plug-in resides.

HTTP method and URI path

```
DELETE /zosmf/externalgateway/system?content=<http-content>
```

where:

- **zosmf/externalgateway** identifies the application server routing services.
- **system** informs the service that the request will be routed to only one application server.
- **content=<*http-content*>** represents the parameters used to qualify the request. Table 11 lists the parameters that are supported for this request.

  Important: If the value for a parameter contains a number sign (#), encode the number sign as *%23*. Otherwise, everything following the number sign will be omitted from the request. For example, if the target is *AppServer#1*, specify *AppServer%231*.

Table 11. Supported input parameters for the application server routing services

Parameter	Required	Description

target	Yes	Nickname assigned to the system entry in the z/OSMF Systems task that describes the settings required to access the application server where the server-side code for your plug-in resides. If the specified system entry does not exist, the request will fail.
		
		z/OSMF stores the nickname for the target application server in the window object in the Browser Object Model. To retrieve the nickname, issue the following JavaScript command from your task:
		`window.frameElement.getAttribute("target")`
		
		For example:
		`postCreate: function() {`
		`  var target = window.frameElement.getAttribute("target");`
		`  var remoteURL = "/zosmf/externalgateway/system?content=`
		`  {'target':'" + target + "','resourcePath':'/testApp'}";`
		`}`

resourcePath	Yes	Path to the service that will process the request.
requestProperties	No	HTTP headers to be included in the HTTP request. Specify the HTTP headers as name and value pairs. If HTTP headers are omitted or are *null*, default values will be used, which are valid for most installations.
timeout	No	Amount of time in milliseconds allowed to process a request. The value can range from 1 to 5601000 milliseconds. If omitted, the default value of 20000 milliseconds is used.
wrapped	No	Indicator of whether the application server routing service will wrap the response from the target application server in a JSON object that contains additional information about the response. Set the parameter to *N* to obtain only the response provided by the target application server. Otherwise, set the parameter to *Y* or omit it to obtain the response along with additional information. For more details, see "Content type used for HTTP response data" on page 32.

Table 11. Supported input parameters for the application server routing services (continued)

Parameter	Required	Description
binary	No	Indicator of whether the response from the target application server is in binary format. Set the parameter to *N* or omit it if the response is not in binary format. Otherwise, set the parameter to *Y* if the response is in binary format.
content	Yes if the HTTP method is POST or PUT.	Parameters or JSON object to include in the body of the HTTPS request that will be sent to the service that will process the request.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Application server routing services" on page 31.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 33.

The response also includes a JSON object that contains the requested information. For more details, see "Content type used for HTTP response data" on page 32.

Example

To remove *objectA* from the application server identified in system entry *appServer1*, which is defined in the z/OSMF Systems task, submit the following request:

```
DELETE /zosmf/externalgateway/system?content={"target":"appServer1","resourcePath":"/testApp/objectA",
"timeout":"30000"} HTTP/1.1

Host: appname.yourco.com
```

Figure 19. Sample request to delete data from an application server

A sample response is shown in Figure 20 on page 45.

```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":
    {
      "systemOutput":
        {
          "result":"success"
        },
      "rc":"Ok",
      "error":null,
     "systemName":"appServer1"
    },
  "numOfSystems":1
}
```

Figure 20. Sample response for deleting data from an application server

Cloud provisioning services

The cloud provisioning services are a set of application programming interfaces (APIs), which are implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to perform software provisioning for IBM Cloud Provisioning and Management for z/OS. This includes creating instances of IBM middleware, such as IBM Customer Information Control System (CICS®), IBM DB2®, IBM Information Management System (IMS™), IBM MQ, and IBM WebSphere Application Server (WAS), and creating middleware resources, such as MQ queues, CICS regions, and DB2 databases..

To exploit cloud provisioning, users known as service providers make software services available to users known as consumers. Consumers, from a selection of software services, can quickly provision and deprovision software services as needed.

To make software services available to consumers, service providers prepare and publish software services templates. The templates guide the provisioning. For example, a template might request that a DB2 subsystem be deployed onto a z/OS system with three databases, or might create a set of CICS AORs and TORs.

To provision the middleware, templates start and run z/OSMF workflows. So, a template includes a workflow definition file, along with other files, which might include a file that defines input variables for the workflow, and a file that defines actions that can be used against the provisioned software.

Software that is provisioned from a template is known as a software services instance. (Note that this is different than a software instance, which you manage with the Software Management task.) You manage a software services instance by using actions such as **Remove** and **deprovision**.

Software services templates are associated with a domain, which defines the associated system. Owners of computing resources, known as landlords, define the domains and the domain administrators. The domain administrator role overlaps with the service provider role. In fact the domain administrator might be a service provider viewed from the standpoint of resource management.

In addition to defining domains, resource management includes defining users for a domain, known as tenants, as well as specifying various administrators.

Network and workload administrators are needed when a template requires a resource pool, that is, a set of z/OS resources that are required by the z/OS software service. When service providers associate templates with tenants, they also indicate whether the template requires a resource pool. The network and workload administrators, then, use the appropriate z/OSMF tasks to complete the resource pool.

In summary, to provision and manage provisioned software, you work with:

Domain
: Defines the management scope for tenants, services, and resource pools.

  A domain consists of a z/OS system. A z/OS system can be in a single domain, or in multiple domains that are managed by a single z/OSMF. There can be up to 36 cloud domains. Cloud domains are defined by landlords. Each cloud domain is assigned one or more domain administrators.

Resource pool
: Identifies z/OS resources that are required by a z/OS software service. A resource pool defines the scope of shared z/OS resources within a cloud domain that has multiple tenants.

Tenant
: Defines the resource sharing scope, for example, a line of business or a class of users.

A tenant consists of a user or group of users that have contracted for use of specified services, and pooled z/OS resources that are associated with the services in a domain.

Instance, or software services instance
Represents software that has been provisioned, typically through the use of templates.

Template, or software services template
Represents a z/OS middleware or a z/OS middleware resource service. A template consists of workflows and input variables that can be used to provision z/OS software, actions that can be used with the provisioned software (the instance), and documentation.

The following are the key roles for IBM Cloud Provisioning and Management for z/OS.

Consumer
A user who has access to the software services and resource pools for a tenant. This access is granted when a domain administrator adds the user's ID to a tenant. A consumer can provision a software services instance, using a software services template, and can manage the lifecycle of a software services instance.

Domain administrator
A user who manages a domain. The domain administrator is responsible for defining software services, tenants, and resource pools for the domain, and managing the relationship across tenants, services, and resource pools.

Landlord
A user who defines the domain, domain administrators, and the associated system resources, for the cloud.

Resource pool administrator
A user who is responsible for managing a resource pool.

Service provider
A user who makes software services available to consumers, by preparing and publishing software services templates.

The basic procedure for provisioning software is:

1. Define domains and tenants. See "Resource management services" on page 70.
2. Create a template, specifying the workflow, action and variables files that were provided by the software vendor.

   The template is added to the software services catalog.
3. Add the template to a tenant.
4. Modify the template as needed.
5. Approve any approval records. Approval records are created when a workflow or action definition file contains an element that identifies a user ID under which a workflow step or action is to be performed (a runAsUser ID). They can also be defined for the template in general, and for a domain.
6. Test the template and ensure that it successfully creates an instance, that is, that it provisions the software and that the actions defined for the instance perform as expected.
7. Publish the template to make it available to consumers.
8. Run the template to create a software instance.

Authorization requirements

Use of the cloud provisioning services APIs requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

In addition, the user's z/OS user ID may need access to other resources, including those that define roles. The specific requirements for each cloud provisioning service are described in the topic for that service. For a summary of resources related to roles, see Table 12.

Table 12. SAF resources for Cloud Provisioning Roles

Role	Class	Resources	Access
Landlord	ZMFCLOUD	<*SAF-prefix*>.ZOSMF.PROVISIONING. RESOURCE_MANAGEMENT.*saf_cloud_groupID_prefix*	READ
Domain administer	ZMFCLOUD	<*SAF-prefix*>.ZOSMF.PROVISIONING. RESOURCE_MANAGEMENT.*domainGroupID*	READ
Domain approver	ZMFCLOUD	<*SAF-prefix*>.ZOSMF.TEMPLATE. APPROVERS.*domainGroupID*	READ
Template approver	ZMFCLOUD	<*SAF-prefix*>.ZOSMF.TEMPLATE. APPROVERS.*domainGroupID.templateID*	READ
Tenant	ZMFCLOUD	<*SAF-prefix*>.ZOSMF.PROVISIONING. RESOURCE_MANAGEMENT.*tenantGroupID*	READ
Resource pool network adminstrator	ZMFCLOUD	<*SAF-prefix*>.ZOSMF. RESOURCE_POOL.NETWORK.*domainGroupID*	READ
Resource pool WLM adminstrator	ZMFCLOUD	<*SAF-prefix*>.ZOSMF. RESOURCE_POOL.WLM.*domainGroupID*	READ

For details about security for the cloud provisioning roles, see *IBM z/OS Management Facility Configuration Guide*.

For information about how to prepare software for provisioning through the REST APIs or the z/OSMF Cloud Provisioning tasks, including the format of the file for defining actions, see Chapter 4, "Preparing software for cloud provisioning," on page 743.

Resource pool services

The resource pool services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to obtain and release network or WLM resources from the network or WLM resource pool that was defined in support of IBM Cloud Provisioning and Management for z/OS. These REST services are intended to be invoked from a workflow step during cloud provisioning and are not available for general use outside of the scope of cloud provisioning.

Table 13 lists the operations that the resource pool services provide.

Resource pool services

Table 13. z/OSMF resource pool services: operations summary

Operation name	HTTP method and URI path
"Obtain an IP address" on page 50	`POST /zosmf/resource-mgmt/rest/<version>/rdp/network/ip/obtain`
"Release an IP address" on page 53	`POST /zosmf/resource-mgmt/rest/<version>/rdp/network/ip/release`
"Obtain a port" on page 55	`POST /zosmf/resource-mgmt/rest/<version>/rdp/network/port/obtain`
"Release a port" on page 58	`POST /zosmf/resource-mgmt/rest/<version>/rdp/network/port/release`
"Obtain a SNA application name" on page 60	`POST /zosmf/resource-mgmt/rest/<version>/rdp/network/snaapplname/actions/obtain`
"Release a SNA application name" on page 63	`POST /zosmf/resource-mgmt/rest/<version>/rdp/network/snaapplname/release`
"Add a classification rule" on page 65	`POST /zosmf/resource-mgmt/rest/<version>/rdp/wlm/clrule/actions/add`
"Remove a classification rule" on page 68	`POST /zosmf/resource-mgmt/rest/<version>/rdp/wlm/classification-rule/actions/remove`

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 204 No Content
The request succeeded.

HTTP 500 Server error
The server encountered an error when it processed the request.

Obtain an IP address

Use this operation to obtain an IP address from a resource pool that has a configured network resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/network/ip/obtain
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation obtains an IP address from a resource pool with a configured network resource pool.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in an IP address being obtained.

Request content

The request content is expected to contain a JSON object that describes the IP address to be obtained. See Table 14.

Table 14. Request content for the obtain IP address request

Field name	Type	Required or optional	Description
template-uuid	String	Required	Unique identifier for the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateID}.
template-name	String	Required	Name of the template that is associated with the resource pool. Derived from workflow internal variable, ${_workflow-templateName}.
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from workflow internal variable, ${_workflow-tenantID}.
network-parms	JSON object	Required	Network. parameters for the request. See Table 15.

Table 15. Network parameters fields

Field	Type	Required or optional	Description
name	String	Optional	Name used in a panel for the Configuration Assistant task. Derived from a workflow internal variable, ${_workflow-softwareServiceInstanceName}.

Table 15. Network parameters fields (continued)

Field	Type	Required or optional	Description
usage-type	String	Optional	Used as a filter. If not specified, only network resource pools without a usage type match. If specified, must match the usage type in the network resource pool definition in the Configuration Assistant task.
ipaddr	String	Required	IP address. The value can be: **A specific IP address** Provision that IP address. The IP address must fit within the available IP address allocation range. Available ranges are associated with the targeted network resource pool, and match the provided usage type. **any4** Provision any available Ipv4 address, from the available range. **any6** Provision any available IPv6 address, from the available range.
system-name	String	See description	Specifies the target system that the resource will be provisioned on. Required if there is more than one system in the network resource pool. Derived from a workflow internal variable, ${_workflow-systemName}.
deployment-id	String	Optional	Workflow-defined string token, used to group all provisioned resources with a server instance.
host-name	String	Optional	Indicates that the domain name server that is configured to the associated IP address allocation range is to be updated with the IP address and the concatenation of the host name and the zone name from the IP address allocation range. Requires a domain name server object to be configured by the network administrator.

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned, with a response body. See "Response content" on page 52.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 16. HTTP error response codes for an obtain IP address request

HTTP error status code	Description
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the request. See Table 17.

Table 17. Response from an obtain IP address request

Field	Type	Description
id	String	Identifier of the IP address.
ipaddr	String	IP address returned from the Configuration Assistant task.

Example HTTP interaction

In Figure 21, a request is submitted to obtain an IP address.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/network/ip/obtain

{
     "template-uuid":"F0F1A1C2",
     "template-name":"CICSBasic",
     "tenant-id":"IZU$0AA",
     "network-parms" :
     {
             "name":"CICSA IP",
             "usage-type":"internal",
             "ipaddr":"any4",
             "system-name":"SY1",
             "deployment-id":"CICSBasic",
             "host-name":"myHostName",
         }
 }
```

Figure 21. Sample request to obtain an IP address, with the request body

The following is the response body for the example obtain IP address request.

```
{
   "id":"101",
   "ipaddr":"192.168.1.1"
 }
```

Release an IP address

Use this operation to release an IP address from a network resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/network/ip/release
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation releases an IP address from a network resource pool, calling through the tenant's resource pool.

On successful completion, HTTP status code `204` (No content) is returned, indicating that the request resulted in an IP address being released.

Request content

The request content is expected to contain a JSON object that describes the IP address to be released. See Table 18.

Table 18. Request content for the release IP address request

Field name	Type	Required or optional	Description
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from workflow internal variable, ${_workflow-tenantID}.
network-parms	JSON object	Required	Network parameters for the request. See Table 19.

Table 19. Network parameters fields

Field	Type	Required or optional	Description
ip-id	String	Required	Identifier of the IP address. This is returned as the id property in an Obtain an IP address request.

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code 204 (No content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 20. HTTP error response codes for a release IP address request

HTTP error status code	Description
HTTP 500 Internal server error	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

None.

Example HTTP interaction

In Figure 22, a request is submitted to release an IP address.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/network/ip/release

{
     "tenant-id":"IYU0AA",
     "network-parms":
     {
           "ip-id":"1001"
         }
 }
```

Figure 22. Sample request to release an IP address, with the request body

Obtain a port

Use this operation to obtain a port from a resource pool that has a configured network resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/network/port/obtain
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation obtains a port from a resource pool with a configured network resource pool.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a port being obtained.

Request content

The request content is expected to contain a JSON object that describes the port to be obtained. See Table 21.

Table 21. Request content for the obtain port request

Field name	Type	Required or optional	Description
template-uuid	String	Required	Unique identifier for the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateID}.
template-name	String	Required	Name of the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateName}.
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-tenantID}.
network-parms	JSON object	Required	Network parameters for the request. See Table 22.

Table 22. Network parameters fields

Field	Type	Required or optional	Description
name	String	Optional	Name, used in a panel for the Configuration Assistant task. Derived from a workflow internal variable, ${_workflow-softwareServiceInstanceName}.

Table 22. Network parameters fields (continued)

Field	Type	Required or optional	Description
usage-type	String	Optional	Used as a filter. If not specified, only network resource pools without a usage type match. If specified, must match the usage type in the network resource pool definition in the Configuration Assistant task.
port	String	Optional	Request port number. If port is not specified, a port is provisioned from available port allocation ranges of the specified transport. If port is specified, it must fit within an available range. Available ranges are those which are associated with the targeted network resource pool and match the provided usage type.
job-name	String	Required	Job name associated with the provisioned instance.
system-name	String	Optional	System name. Derived from a workflow internal variable, ${_workflow-systemName}.
deployment-id	String	Optional	Workflow-defined string token, used to group all provisioned resources with a server instance.
host-name	String	Optional	Host-name for Configuration Assistant.

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned, with a response body. See "Response content."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 23. HTTP error response codes for an obtain port request

HTTP error status code	Description
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the request. See Table 24.

Table 24. Response from an obtain port request

Field	Type	Description
id	String	Identifier of the port.
port	String	Port number.

Example HTTP interaction

In Figure 23, a request is submitted to obtain a port.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/network/port/obtain

{
     "template-uuid":"F0F1A1C2",
     "template-name":"CICSBasic",
     "tenant-id":"IZU$0AA",
     "network-parms" :
     {
      "name":"PortForCics1",
      "port":"80",
      "usage-type":"Internal",
           "job-name":"WLP001"
      "deployment-id":"CICSBasic",
      "system-name":"SY1",
      "host-name":"myHostName",
      }
 }
```

Figure 23. Sample request to obtain a port, with request body

The following is the response body for the example obtain port request.

```
{
  "id":82346
  "port":"80",
 }
```

Release a port

Use this operation to release a port from a network resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/network/port/release
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation releases a port from a network resource pool, calling through the tenant's resource pool.

On successful completion, HTTP status code `204` (No content) is returned, indicating that the request resulted in a port being released.

Request content

The request content is expected to contain a JSON object that describes the port to be released. See Table 25.

Table 25. Request content for the release port request

Field name	Type	Required or optional	Description
template-uuid	String	Required	Unique identifier for the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateID}.
template-name	String	Required	Name of the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateName}.
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-tenantID}.
network-parms	JSON object	Required	Network parameters for the request. See Table 26.

Table 26. Network parameters fields

Field	Type	Required or optional	Description
port-id	String	Required	Identifier of the port. This is returned as the id property in an Obtain a port request.

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (No content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 27. HTTP error response codes for a release port request

HTTP error status code	Description
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

None.

Example HTTP interaction

In Figure 24, a request is submitted to release a port.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/network/port/release

{
     "template-uuid":"F0F1A1C2",
     "template-name":"CICSBasic",
     "tenant-id":"IYU0AA",
     "network-parms" :
     {
           "port-id" : "1001"
        }
 }
```

Figure 24. Sample request to release a port, with the request body

Obtain a SNA application name

Use this operation to obtain a SNA application name from a resource pool that has a configured network resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/network/snaapplname/actions/obtain
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation obtains a SNA application name from a resource pool with a configured network resource pool.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a SNA application name being obtained.

Request content

The request content is expected to contain a JSON object that describes the SNA application name to be obtained. See Table 28.

Table 28. Request content for the obtain SNA application name request

Field name	Type	Required or optional	Description
template-uuid	String	Required	Unique identifier for the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateID}.
template-name	String	Required	Name of the template that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-templateName}.
tenant-id	String	Required	Name of the tenant that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-tenantID}.
network-parms	Array	Required	Network parameters for the request. See Table 29.

Table 29. Network parameters fields

Field	Type	Optional/ Required	Description
name	String	Optional	Name, used in a panel for the Configuration Assistant task.
deployment-id	String	Optional	Workflow-defined string token.

Table 29. Network parameters fields (continued)

Field	Type	Optional/ Required	Description
sna-appl-name	String	Required	A name for the SNA application. Derived from the workflow internal variable ${_workflow-softwareServiceInstanceName}.

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned, with a response body. See "Response content."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 30. HTTP error response codes for an obtain SNA application name request

HTTP error status code	Description
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the request. See Table 31.

Table 31. Response from an obtain SNA application name request

Field	Type	Description
id	String	Identifier of the SNA application name. Needed for the release call, as the value for the appl-name-id property.
appl-name	String	Required. Application name from the network resource pool in the Configuration Assistant task.

Example HTTP interaction

In Figure 25 on page 62, a request is submitted to obtain a SNA application name.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/network/snaapplname/actions/obtain

{
   "template-uuid":"F0F1A1C2",
   "template-name":"CICSBasic",
   "tenant-id":"IZU$0AA",
    "network-parms" :
    {
      "name":"CICSA APPLID",
      "deployment-id":"CICSBasic",
      "sna-appl-name":"CICSA001"
     }
}
```

Figure 25. Sample request to obtain a SNA application name, with request body

The following is the response body for the example obtain SNA application name request.

```
{
       "id":82346,
       "appl-name":"CICSC10"
 }
```

Release a SNA application name

Use this operation to release a SNA application name from a network resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/network/snaapplname/release
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation releases a SNA application name from a network resource pool, calling through the tenant's resource pool.

On successful completion, HTTP status code `204` (No content) is returned, indicating that the request resulted in a SNA application name being released.

Request content

The request content is expected to contain a JSON object that describes the SNA application name to be released. See Table 32.

Table 32. Request content for the release SNA appplication name request

Field name	Type	Required or optional	Description
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-tenantID}.
network-parms	Array	Required	Network parameters for the request. See Table 33.

Table 33. Network parameters fields

Field	Type	Optional/ Required	Description
appl-name-id	String	Required	Identifier of the SNA application name. This is returned as the id property in an Obtain a SNA application name request.

Authorization requirements

The user must be a consumer in the tenant, or a network administrator in the domain that the tenant is associated with.

For more information, see “Authorization requirements” on page 47.

HTTP status codes

On successful completion, HTTP status code 204 (No content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 34. HTTP error response codes for a release SNA application name request

HTTP error status code	Description
HTTP 500 Internal server error	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

None.

Example HTTP interaction

In Figure 26, a request is submitted to release a SNA application name.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/network/snaapplname/release
{
     "tenant-id":"IYU0AA",
     "network-parms":
     {
           "appl-name-id":"82346"
        }
 }
```

Figure 26. Sample request to release a SNA application name

Add a classification rule

Use this operation to add a classification rule in a WLM Policy with service level agreement specified in a resource pool.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/wlm/clrule/actions/add
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation adds a classification rule in a WLM policy for a middleware instance with a service level agreement specified in the tenant's resource pool.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a classification rule being added.

Request content

The request content is expected to contain a JSON object. See Table 35.

Table 35. Request content for the add classification rule request

Field name	Type	Required or optional	Description
template-name	String	Required	Name of the template that is associated with the tenant. Derived from a workflow internal variable, ${_workflow-templateName}.
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-tenantID}.
registry-id	Array	Required	ID of the software services registry. Derived from a workflow internal variable, ${_workflow-registryID}.See Table 36.

Table 36. WLM parameters fields

Field	Type	Optional/ Required	Description
qualifier	String	Required	The started task name. In most cases it can be derived from workflow internal variable ${_workflow-softwareServiceInstanceName}

Authorization requirements

The user must be a consumer in the tenant, or a WLM administrator in the domain that the tenant is associated with.

The user must also be the owner of the software services registry entry for the software services instance.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 37. HTTP error response codes for a release SNA application name request

HTTP error status code	Description
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the request. See Table 38.

Table 38. Response from an add classification rule request

Field	Type	Description
cl-rule-id	String	Required. Identifier of the classification rule.
report-class-name	String	Report class that is associated with the resource pool.
service-class-name	String	Service class that is associated with the SLA that is defined in the resource pool.

Example HTTP interaction

In Figure 27, a request is submitted to add a classification rule.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/wlm/clrule/actions/add
{
   "template-name":"CICSBasic",
   "tenant-id":"IYU102",
   "resistry-id":"0d375584-305d-4bd5-b26e-88ac74c8171a",
   "wlm-parms":
    {
      "qualifier":"CICSA001"
    }
}
```

Figure 27. Sample request to add a classification rule

The following is the response body for the request:

```
|  {
|       "cl-rule-id" : "82346",
|       "report-class-name":"RPTCLASS",
|       "service-class-name":"SCGOLD",
|  }

|
```

Remove a classification rule

Use this operation to remove a classification rule from a WLM Policy.

HTTP method and URI path

```
POST /zosmf/resource-mgmt/rest/<version>/rdp/wlm/classification-rule/actions/remove
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation removes a classification rule from a WLM policy.

On successful completion, HTTP status code `204` (No content) is returned, indicating that the request resulted in a classification rule being renoved.

Request content

The request content is expected to contain a JSON object. See Table 39.

Table 39. Request content for the remove classification rule request

Field name	Type	Required or optional	Description
template-name	String	Required	Name of the template that is associated with the tenant. Derived from a workflow internal variable, ${_workflow-templateName}.
tenant-id	String	Required	ID of the tenant that is associated with the resource pool. Derived from a workflow internal variable, ${_workflow-tenantID}.
registry-id	Array	Required	ID of the software services registry. Derived from a workflow internal variable, ${_workflow-registryID}.See Table 40.

Table 40. WLM parameters fields

Field	Type	Optional/ Required	Description
cl-rule-id	String	Required	Returned by an Add Classification Rule request as the id property.

Authorization requirements

The user must be a consumer in the tenant, or a WLM administrator in the domain that the tenant is associated with.

The user must also be the owner of the software services registry entry for the software services instance.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (No Content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 41. HTTP error response codes for a release SNA application name request

HTTP error status code	Description
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

None.

Example HTTP interaction

In Figure 28, a request is submitted to add a classification rule.

```
POST https://localhost:4444/zosmf/resource-mgmt/rest/1.0/rdp/wlm/classification-rule/actions/remove
{
   "template-name":"CICSBasic",
   "tenant-id":"IYU102",
   "registry-id":"0d375584-305d-4bd5-b26e-88ac74c8171a",
   "wlm-parms":
    {
       "cl-rule-id" : 82346,
    }
}
```

Figure 28. Sample request to remove a classification rule

Resource management services

The resource management services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to get and list domains and tenants that were defined in support of IBM Cloud Provisioning and Management for z/OS.

Table 42 lists the operations that the resource management services provide.

Resource management services

Table 42. z/OSMF resource management services: operations summary

Operation name	HTTP method and URI path
"Get a domain" on page 71	`GET /zosmf/resource-mgmt/rest/<version>/domains/<object-id>`
"List the domains" on page 74	`GET /zosmf/resource-mgmt/rest/<version>/domains/`
"Get a tenant" on page 77	`GET /zosmf/resource-mgmt/rest/<version>/tenants/<object-id>`
"List the tenants" on page 79	`GET /zosmf/resource-mgmt/rest/<version>/tenants/`

Authorization requirements

Use of the Resource Management services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

In addition, the user's z/OS user ID may need access to other resources, including those that define roles such as the landlord and domain administrator. The specific requirements for each resource management service are described in the topic for that service. For an overview of the security requirements for cloud provisioning roles, see "Authorization requirements" on page 47. For details, see *IBM z/OS Management Facility Configuration Guide*. .

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 400 Bad request
There is a missing field in the request body.

HTTP 404 Not found
The requested resource does not exist.

HTTP 500 Server error
The server encountered an error when it processed the request.

Get a domain

Use this operation to retrieve a domain.

HTTP method and URI path

```
GET /zosmf/resource-mgmt/rest/<version>/domains/<object-id>
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the domain to be retrieved.

Query parameters

None.

Description

This operation retrieves a domain.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a domain being retrieved.

Request content

None.

Authorization requirements

The user must be a landlord, domain administrator, or consumer in the domain.

For more information, see "Resource management services" on page 70.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned, and with a response body. See "Response content" on page 72.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 43. HTTP error response codes for a get domain request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contains incorrect parameters.
`HTTP 404 Not found`	The requested domain does not exist.
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the domain. See Table 44.

Table 44. Response from a get domain request

Field	Type	Description
domain-id	String	The generated ID for the domain.
domain-name	String	Descriptive name for the domain.
domain-system-list	Array	Array describing the systems in the domain. See Table 45.
domain-administrator-list	String	List of user IDs for the domain administrators.
network-administrator-list	String	List of user IDs for the network administrators.
wlm-administrator-list	String	List of user IDs for the WLM administrators.
security-administrator	String	User ID of the security administrator.
security-job-statement	String	JOB statement JCL used in security jobs for the domain.
domain-approver-list	String	List of user IDs for the domain approvers.
object-uri	String	URI of the newly created object.
domain-description	String	Description of the domain.

Table 45. Response from a get request: Systems

Field	Type	Description
sysplex-name	String	Name of the sysplex. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.
sysplex-node-name	String	Sysplex node name.
system-nickname	String	Unique name that is assigned to the system definition.

Example HTTP interaction

In Figure 29, a request is submitted to retrieve a domain.

```
GET https://localhost:4444/zosmf/resource-mgmt/rest/1.0/domains/<object-id>
```

Figure 29. Sample request to get a domain

The following is the response body for the example get domain request.

```
{
          "domain-id":"izu$0",
          "domain-name":"default",
          "domain-system-list":[{
              "sysplex-name":"DUMBPLEX",
              "sysplex-node-name":"DUMBNODE",
              "system-nickname":"DUMBNODE_001"},
          ...
          ],
```

```
|           "domain-administrator-list":["ZOSMFT1", ... ],
|           "network-administrator-list":["ZOSMFT1", ... ],
|           "wlm-administrator-list":["ZOSMFT1", ... ],
|           "security-administrator":"ZOSMFT1",
|           "security-job-statement" : "//JOBCARD JOB(acct-info)",
|           "domain-approver-list":["ZOSMFT1", ... ],
|           "object-uri":"/zosmf/resource-mgmt/rest/1.0/domains/izu$0",
|           "domain-description":"default domain"
|       }

|
```

List the domains

Use this operation to list the domains that are defined for IBM Cloud Provisioning and Management for z/OS.

HTTP method and URI path

```
GET /zosmf/resource-mgmt/rest/<version>/domains/
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation lists the domains for cloud provisioning.

On successful completion, HTTP status code `200` (OK) is returned, and a response body is returned. See "Response content" on page 75.

Request content

None.

Authorization requirements

The user must be a landlord, domain administrator, or consumer in the domain.

For more information, see "Resource management services" on page 70.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 46. HTTP error response codes for a list domains request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contains incorrect parameters.
`HTTP 404 Not found`	The requested domain does not exist.
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the domains. See Table 47.

Table 47. Response from a list domains request

Field	Type	Description
domain-list	Array	Domains. See Table 48.
domain-administrator-list	Array	User IDs for the domain administrators.
network-administrator-list	Array	User IDs for the network administrators.
wlm-administrator-list	Array	User IDs for the WLM administrators.
security-administrator	Array	User IDs for the security administrators.
security-job-statement	String	JOB statement JCL used in security jobs for the domain.
domain-approver-list	Array	User IDs for the domain approvers.
object-uri	String	URI of the newly created object.
domain-description	String	Description of the domain.

Table 48. Response from a list domains request: domains

Field	Type	Description
domain-id	String	The generated ID for the domain.
domain-name	String	Descriptive name for the domain.
domain-system-list	Array	Systems in the domain. See Table 49.

Table 49. Response from a list domains request: systems

Field	Type	Description
sysplex-name	String	Name of the sysplex. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.
sysplex-node-name	String	Sysplex node name.
system-nickname	String	Unique name that is assigned to the system definition.

Example HTTP interaction

In Figure 30, a request is submitted to list the domains.

```
GET https://localhost:4444/zosmf/resource-mgmt/rest/1.0/domains/
```

Figure 30. Sample request to list domains

The following is the response body for the example list domains request.

```
{
     "domain-list": [{
         "domain-id": "izu$0",
```

```
|           "domain-name": "default",
|           "domain-system-list": [{
|               "sysplex-name": "DUMBPLEX",
|               "sysplex-node-name": "DUMBNODE",
|               "system-nickname": "DUMBNODE_001"},
|           ...
|           ],
|           "domain-administrator-list": ["ZOSMFT1", ... ],
|           "network-administrator-list": ["ZOSMFT1", ... ],
|           "wlm-administrator-list": ["ZOSMFT1", ... ],
|           "security-administrator":"ZOSMFT1",
|           "security-job-statement" : "//JOBCARD JOB(acct-info)",
|           "domain-approver-list": ["ZOSMFT1", ... ],
|           "object-uri": "/zosmf/resource-mgmt/rest/1.0/domains/izu$0",
|           "domain-description": "default domain"
|       },
|    ...
|    ]
|   }

|
```

Get a tenant

Use this operation to retrieve a tenant.

HTTP method and URI path

```
GET /zosmf/resource-mgmt/rest/<version>/tenants/<object-id>
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the tenant to be retrieved.

Query parameters

None.

Description

This operation retrieves a tenant.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a tenant being retrieved, and a response body is returned. See "Response content" on page 78.

Request content

None.

Authorization requirements

The user must be a domain administrator, or a consumer in the tenant.

For more information, see "Resource management services" on page 70.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 50. HTTP error response codes for a get tenant request

HTTP error status code	Description
`HTTP 400 Bad request`	The request body is missing a field.
`HTTP 404 Not found`	The requested tenant does not exist.
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the tenant. See Table 51.

Table 51. Response from a get tenant request

Field	Type	Description
tenant-id	String	The generated ID for the tenant.
tenant-name	String	Descriptive name for the tenant.
tenant-domain-id	String	The generated ID for the domain to which the tenant belongs.
tenant-domain-name	String	Descriptive name of the domain to which the tenant belongs.
tenant-templates	Array	Describes the templates for the tenant. See Table 52.
tenant-consumer-list	Array	Consumer user IDs for the tenant.
object-uri	String	URI of the newly created object.
tenant-description	String	Description of the tenant.

Table 52. Response from a get request: tenant-templates

Field	Type	Description
rdp-id	String	Identifier of the resource deployment pool.
template-available	String	Indicates if the template is available (true or false).
template-id	String	Identifier of the template.
template-name	String	Name of the template.

Example HTTP interaction

In Figure 31, a request is submitted to retrieve a tenant.

```
GET https://localhost:4444/zosmf/resource-mgmt/rest/1.0/tenants/<object-id>
```

Figure 31. Sample request to get a tenant

The following is the response body for the example get tenant request.

```
{
   "tenant-id": "izu$000",
   "tenant-name": "default",
   "tenant-domain-id": "izu$0",
   "tenant-domain-name": "default",
   "tenant-templates": [
    {
    "rdp-id": "000",
    "template-available": false,
    "template-id": "template-id0",
    "template-name": "tem0 name"
    }, ...
   ],
   "tenant-consumer-list": [
    "ZOSMFAD",
    "ZOSMFT2"
   ],
   "object-uri": "\/zosmf\/resource-mgmt\/rest\/1.0\/tenants\/izu$000",
   "tenant-description": "default tenant",
 }
```

List the tenants

Use this operation to list the tenants that are defined for IBM Cloud Provisioning and Management for z/OS.

HTTP method and URI path

```
GET /zosmf/resource-mgmt/rest/<version>/tenants/
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation lists the tenants that are defined for IBM Cloud Provisioning and Management for z/OS.

On successful completion, HTTP status code `200` (OK) is returned, and a response body is returned. See "Response content."

Request content

None.

Authorization requirements

The user must be the domain administrator, or a consumer in the tenant.

For more information, see "Resource management services" on page 70.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 53. HTTP error response codes for a list tenants request

HTTP error status code	Description
`HTTP 404 Not found`	The domain does not exist.
`HTTP 500 Internal server error`	The server encountered an error. See the response body for a JSON object with information about the error.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the tenants. See Table 54 on page 80.

Table 54. Response from a list tenants request

Field	Type	Description
tenant-list	Array	Information about the tenants that are defined. See Table 55.

Table 55. Response from a list tenants request: Tenants

Field	Type	Description
tenant-id	String	The generated ID for the tenant.
tenant-name	String	Descriptive name for the tenant.
tenant-domain-id	String	Identifier of the domain to which the tenant belongs.
tenant-domain-name	String	Descriptive name of the domain to which the tenant belongs.
tenant-templates	Array	Describes the templates for the tenant. See Table 56.
tenant-consumer-list	Array	User IDs for the consumers in the tenant.
object-uri	String	URI of the newly created object.
tenant-description	String	Description of the tenant.

Table 56. Response from a list tenants request: Templates

Field	Type	Description
template-available	String	Indicates if the template is available (true or false).
template-id	String	Identifier of the template.
template-name	String	Name of the template.

Example HTTP interaction

In Figure 32, a request is submitted to list the tenants.

```
GET https://localhost:4444/zosmf/resource-mgmt/rest/1.0/tenants/
```

Figure 32. Sample request to list tenants

The following is the response body for the example list tenants request.

```
{
   "tenant-List" : [{
       "tenant-id": "izu$000",
        "tenant-name": "default",
  "tenant-domain-id": "izu$0",
  "tenant-domain-name": "default",
  "tenant-templates": [
   {
    "rdp-id": "000",
    "template-available": false,
    "template-id": "template-id0",
    "template-name": "tem0 name"
   }, ...
   ],
 "tenant-consumer-list": [
    "ZOSMFAD",
    "ZOSMFT2"
     ],
 "object-uri": "\/zosmf\/resource-mgmt\/rest\/1.0\/tenants\/izu$000",
```

```
|   "tenant-description": "default tenant",
|   }, ...
|   ]
|   }

|
```

Software services template services

The software services template services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to create and manage software services templates, which can be used to provision z/OS software in support of IBM Cloud Provisioning and Management for z/OS.

For information about cloud provisioning, including a description of the roles, see "Cloud provisioning services" on page 46.

The basic procedure for provisioning software is:

1. Define domains and tenants. See "Resource management services" on page 70.
2. Create a template, specifying the workflow, action and variables files that were provided by the software vendor.

   The template is added to the software services catalog.
3. Add the template to a tenant.
4. Modify the template as needed.
5. Approve any approval records. Approval records are created when a workflow or action definition file contains an element that identifies a user ID under which a workflow step or action is to be performed (a runAsUser ID). They can also be defined for the template in general, and for a domain.
6. Test the template and ensure that it successfully creates an instance, that is, that it provisions the software and that the actions defined for the instance perform as expected. Optionally, clean up the results of your test, that is, deprovision and remove the instance that you created by testing the template.
7. Publish the template to make it available to consumers.
8. Run the template to create a software instance.

Table 57 lists the operations that the software services template services provide.

"Published software service template services" on page 138 describes the REST APIs for working with published software services templates, for example, for running a template to create an instance.

"Software services instance services" on page 157 describes the REST APIs for working with software services instances.

Software services template

Table 57. z/OSMF software services template services: operations summary

Operation name	HTTP method and URI path
"Create a software services template" on page 86	`POST /zosmf/provisioning/rest/<version>/scc`
"Create a new version of a software services template" on page 89	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/create_new_version`
"Create a new software services template based on an existing one" on page 92	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/create_based_on`
"Modify a software services template" on page 94	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>`

Table 57. z/OSMF software services template services: operations summary (continued)

Operation name	HTTP method and URI path
"Delete a software services template" on page 97	`DELETE /zosmf/provisioning/rest/<version>/scc/<object-id>`
"List the software services templates" on page 111	`GET /zosmf/provisioning/rest/<version>/scc`
"Get a software services template" on page 98	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>`
"Get software services template documentation" on page 104	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/documentation/admin` `GET /zosmf/provisioning/rest/<version>/scc/<object-id>/documentation/consumer`
"Get prompt variables for a software services template" on page 106	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/prompt-variables`
"Get source information for a software services template" on page 109	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/sources`
"Publish a software services template" on page 115	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/publish`
"Test a software services template" on page 117	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/test`
"Refresh a software services template" on page 120	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/refresh`
"Archive a software services template" on page 122	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/archive`
"Add an approval for a software services template" on page 124	`POST /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals`
"Get an approval for a software services template" on page 126	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/<approval-object-id>`
"List the approvals for a software services template" on page 129	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals`
"Approve an approval record for a software services template" on page 132	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/<approval-object-id>/actions/approve`
"Reject the use of your user ID with a software services template" on page 134	`GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/<approval-object-id>/actions/reject`

Table 57. z/OSMF software services template services: operations summary (continued)

Operation name	HTTP method and URI path
"Delete an approval for a software services template" on page 136	`DELETE /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/<approval-object-id>`

Authorization requirements

Use of the software services template services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

In addition, the user's z/OS user ID may need access to other resources, including those that define roles such as the landlord and domain administrator. The specific requirements for each software services template service are described in the topic for that service. For an overview of the security requirements for cloud provisioning roles, see "Authorization requirements" on page 47. For details, see *IBM z/OS Management Facility Configuration Guide*.

Error response content

For the 4*nn* HTTP error status codes, additional diagnostic information beyond the HTTP status code is provided in the response body for the request. This information is provided in the form of a JSON object containing the following fields:

Table 58. Response from a software services template request failure

Field	Type	Description
`http-status`	String	HTTP status code.
`request-method`	String	HTTP request method.
`request-uri`	String	HTTP request URI.
`reason`	String	HTTP status reason code.
`message`	String	Message describing the error.
`detailed-message`	String	Message describing the error in more detail.
`debug`	String	Debug information about for the error.

Error logging

Errors from the software services template services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 201 Created
The request succeeded and resulted in the creation of an object.

HTTP 202 Accepted
The request was successfully validated and is performed asynchronously.

HTTP 204 No content
The request succeeded, but no content is available to be returned.

HTTP 400 Bad request
The request contained incorrect parameters.

HTTP 401 Unauthorized
The request cannot be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both. Or, the client did not authenticate to z/OSMF by using a valid WWW-Authenticate header.

HTTP 404 Not found
The requested resource does not exist.

HTTP 409 Request conflict
The request cannot be processed because of conflict in the request, such as an edit conflict between multiple updates.

Related information

The publish operation locks the template, preventing any further modification, and creates a public copy of it. To work with a published software services template, use the REST APIs that are described in "Published software service template services" on page 138.

The run operation for a published template creates a workflow, starts the workflow, and creates a corresponding software services instance in the software services registry. To work with a software services instance, use the REST APIs described in "Software services instance services" on page 157.

Create a software services template

Use this operation to create a software services template in the catalog. The template is a private entry until it is published.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates a software services template in the catalog, based on the properties that are specified in the request body (a JSON object). For the properties that you can specify, see "Request content."

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the creation of a new software services template. A response body is provided, as described in "Response content" on page 87.

Request content

The request content is expected to contain a JSON object that describes the software services template to be created. See Table 59.

Table 59. Request content for the software services template request

Field name	Type	Required or optional	Description
action-definition-file	String	Required	Location of the action definition file, a file in XML format that defines the actions for the software services instance provisioned from the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name. For example, specify /usr/lpp/zosmf/v2r1/samples/actions.xml.
description	String	Optional	Description of the software services template.
name	String	Required	Descriptive name for the software services template. The name must be unique, no longer than 48 characters, and consist of alphanumeric characters (A-Z, a-z, and 0-9), national characters ($@), underscore (_), and hyphen (-).
workflow-definition-file	String	Required	Location of the workflow definition file, the primary XML file that defines the workflow. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name. For example, specify /usr/lpp/zosmf//v2r1/samples/workflow_sample_automation.xml.

Table 59. Request content for the software services template request (continued)

Field name	Type	Required or optional	Description
workflow-variable-input-file	String	Optional	Location of the workflow variable input file, an optional properties file used to specify in advance the values for one or more of the variables that are defined in the workflow definition file. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
domain-name	String	Optional	Name of the domain. Required if the user ID has administrator authorization to more than one domain.
approvals	Array of strings	Optional	An array of strings representing user IDs of users that are responsible for approving the template.
workflow-clean-after-provisioned	boolean	Optional	Indicates if a workflow that performs provisioning should be automatically deleted after it completes successfully. The value is true to delete, false to keep. The default value, if none is specified, is false, which keeps the workflow.
consumer-documentation-file	String	Optional	Location of a file that provides information for consumers about the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
consumer-documentation-type	String	Optional	Type of the consumer documentation file, either text or pdf. This is required if consumer-documentation-file is specified.
admin-documentation-file	String	Optional	Location of a file that provides information for administrators about the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
admin-documentation-type	String	Optional	Type of the administrator documentation file, either text or pdf. This is required if admin-documentation-file is specified.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content."

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services template. Table 60 on page 88 lists the fields in the JSON object.

Table 60. Response from a create software services template request

Field	Type	Description
generated-name	String	The generated name associated with this software services template.
object-id	String	The object ID of the newly created software services template. The object ID is to be used on further requests to the session.
object-uri	String	The object URI of the newly created software services template.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 61. Response from a software services template request failure

Field	Type	Description
http-status	String	HTTP status code.
request-method	String	HTTP request method.
request-uri	String	HTTP request URI.
reason	String	HTTP status reason code.
message	String	Message describing the error.
detailed-message	String	Message describing the error in more detail.
debug	String	Debug information about for the error.

Example HTTP interaction

The example in Figure 33 shows a request to create a software services template on the system SY1.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/

    {
    "name":"mq",
    "description":"Provision an mq instance.",
    "action-definition-file":"/u/zoscloud-beta/factory/mq/qmgr/qmgrActions.xml",
    "workflow-definition-file":"/u/zoscloud-beta/factory/mq/qmgr/provision.xml",
    "domain-name":"default",
    "workflow-clean-after-provisioned":true,
    "workflow-variable-input-file":"/u/zoscloud-beta/factory/mq/workflow_variables.properties",
    "consumer-documentation-file":"/u/zoscloud-beta/factory/mq/mqaas_readme.pdf",
    "consumer-documentation-type":"pdf",
    "admin-documentation-file":"/u/zoscloud-beta/factory/mq/admin_readme.txt",
    "admin-documentation-type":"text",
    "approvals":["zosmfad"]
    }
```

Figure 33. Sample request to create a software services template

The following is the response body for the request.

```
{
        "generated-name": "mqDemo.1.default",
        "object-id": "a79cbb30-f960-403b-8dd3-cccb44b9f192",
        "object-uri": "/zosmf/provisioning/rest/1.0/scc/a79cbb30-f960-403b-8dd3-cccb44b9f192"
}
```

Figure 34. Sample response body

Create a new version of a software services template

You can use this operation to create a new version of a software services template, with the same name as the original, associated with the same domain and tenants, but with new source files.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/create_new_version
```

In this request

<object-id>
Identifies the existing software services template to create a new version of.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates a new version of an existing software services template in the catalog. The new version has the same name as the original entry, and is associated with the same domain and tenants. However, it has new source files (workflow definition, action definition, variable input, and documentation). You cannot already have a draft software services template of this version.

The new version is assigned a version number that is the next available number in sequence.

The template that you create a new version of must be in the published or archived state.

On successful completion, HTTP status code `201` (Normal) is returned, indicating that the request resulted in the creation of a new version of a software services template. A response body is provided, as described in "Response content" on page 91.

Request content

The request content is expected to contain a JSON object that describes the software services template to be created. See Table 62.

Table 62. Request content for the software services template request

Field name	Type	Required or optional	Description
action-definition-file	String	Required	Location of the action definition file, a file in XML format that defines the actions for the software services instance provisioned from the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name. For example, specify /usr/lpp/zosmf/v2r1/samples/actions.xml.
description	String	Optional	Description of the software services template, up to 500 characters.

Table 62. Request content for the software services template request (continued)

Field name	Type	Required or optional	Description
workflow-definition-file	String	Required	Location of the workflow definition file, the primary XML file that defines the workflow. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name. For example, specify /usr/lpp/zosmf//v2r1/samples/workflow_sample_automation.xml.
workflow-variable-input-file	String	Optional	Location of the workflow variable input file, an optional properties file used to specify in advance the values for one or more of the variables that are defined in the workflow definition file. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
approvals	Array of strings	Optional	An array of strings representing user IDs of users that are responsible for approving the template.
workflow-clean-after-provisioned	boolean	Optional	Indicates if a workflow that performs provisioning should be automatically deleted after it completes successfully. The value is true to delete, false to keep. The default value, if none is specified, is false, which keeps the workflow.
consumer-documentation-file	String	Optional	Location of a file that provides information for consumers about the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
consumer-documentation-type	String	Optional	Type of the consumer documentation file, either text or pdf. This is required if consumer-documentation-file is specified.
admin-documentation-file	String	Optional	Location of a file that provides information for administrators about the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
admin-documentation-type	String	Optional	Type of the administrator documentation file, either text or pdf. This is required if admin-documentation-file is specified.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content" on page 91.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services template. Table 63 lists the fields in the JSON object.

Table 63. Response from a create new version of a software services template request

Field	Type	Description
generated-name	String	The generated name associated with this software services template.
object-id	String	The object ID of the newly created software services template. The object ID is to be used on further requests to the session.
object-uri	String	The object URI of the newly created software services template.

If a failure occurs, the response body contains a JSON object with a description of the error.

Example HTTP interaction

In Figure 35, a request is submitted to create a new version of a software services template on the system SY1.

```
POST /zosmf/provisioning/rest/1.0/scc/0389ed37-fe13-4176-af65-c171b6ba6b37/actions/
create_new_version HTTP/1.1
```

Figure 35. Sample request to create a new version of a software services template

```
{
generated-name: "mq.2.default"
object-id: "cd00fb41-20ed-4133-b985-52e28edfcfd0"
object-uri: "/zosmf/provisioning/rest/1.0/scc/cd00fb41-20ed-4133-b985-52e28edfcfd0"
}
```

Figure 36. Sample response body

Create a new software services template based on an existing one

You can use this operation to create a new software services template based on one that already exists, with the same source files.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/create_based_on
```

In this request

<object-id>
Identifies the existing software services template.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates a new software services template in the catalog, based on the existing software services template identified by the object ID. It has the same source files (workflow definition, action definition, variable input, and documentation).

On successful completion, HTTP status code `201` (Normal) is returned, indicating that the request resulted in the creation of a new version of a software services template. A response body is provided, as described in "Response content" on page 93.

Request content

The request content is expected to contain a JSON object. See Table 64.

Table 64. Request content for the software services template request

Field name	Type	Required or optional	Description
name	String	Required	Descriptive name for the software services template. The name must be unique, no longer than 48 characters, and consist of alphanumeric characters (A-Z, a-z, and 0-9), national characters ($@), underscore (_), and hyphen (-).
domain-name	String	Varies	Name of the domain. Required if the user ID has administrator privileges to more than one domain.
approvals	Array of strings	Optional	An array of strings representing the user IDs that are responsible for approving the template.
target-copy-path	String	Required	The absolute path name of an empty z/OS UNIX directory. The source file contents of the existing software services template are copied into this location, and the new template is created based on that content. If the directory does not exist, it is created. However, the parent directory must already exist.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content."

Response content

On successful completion, the service returns a response body, which contains a JSON object. Table 65 lists the fields in the JSON object.

Table 65. Response from a successful request

Field	Type	Description
`generated-name`	String	The generated name associated with this software services template.
`object-id`	String	The object ID of the newly created software services template. The object ID is to be used on further requests to the session.
`object-uri`	String	The object URI of the newly created software services template.

If a failure occurs, the response body contains a JSON object with a description of the error.

Example HTTP interaction

In Figure 37, a request is submitted to create a new version of a software services template on the system `SY1`.

```
POST /zosmf/provisioning/rest/1.0/scc/0389ed37-fe13-4176-af65-c171b6ba6b37/actions/create_based_on
    HTTP/1.1

{
"name" : "config2",
"target-copy-path":"/users/gg/zosmf/newConfig2"
}
```

Figure 37. Sample request to create a new software services template based on an existing one, with request body

```
{
    generated-name: "mqUpgrade.1.default"
    object-id: "cd00fb41-20ed-4133-b985-52e28edfcfd0"
    object-uri: "/zosmf/provisioning/rest/1.0/scc/cd00fb41-20ed-4133-b985-52e28edfcfd0"
}
```

Figure 38. Sample response body

Modify a software services template

You can use this operation to modify fields in a software services template in the catalog.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>
```

In this request

<object-id>
Identifies the software services template to be modified.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation modifies fields in a software services template in the catalog, based on the properties that are specified in the request body (a JSON object). For the properties that you can specify, see "Request content."

On successful completion, HTTP status code `204` (Normal) is returned, indicating that the request resulted in a modified software services template.

The software services template must be in one of the draft states.

Modifying any of the definition files causes all approvals to be reset.

Request content

The request content is expected to contain a JSON object that describes the fields to be modified. See Request content for the software services template request.

Table 66. Request content for the software services template request

Field name	Type	Required or optional	Description
action-definition-file	String	Optional	Location of the action definition file, a file in XML format that defines the actions for the software services instance provisioned from the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name. For example, specify /usr/lpp/zosmf/v2r1/samples/actions.xml.
description	String	Optional	Description of the software services template.
workflow-definition-file	String	Optional	Location of the workflow definition file, the primary XML file that defines the workflow. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name. For example, specify /usr/lpp/zosmf//v2r1/samples/workflow_sample_automation.xml.

Table 66. Request content for the software services template request (continued)

Field name	Type	Required or optional	Description
workflow-variable-input-file	String	Optional	Location of the workflow variable input file, an optional properties file used to specify in advance the values for one or more of the variables that are defined in the workflow definition file. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
workflow-clean-after-provisioned	Boolean	Optional	Indicates if a workflow that performs provisioning should be automatically deleted after it completes successfully. The value is true to delete, false to keep. The default value, if none is specified, is false, which keeps the workflow.
consumer-documentation-file	String	Optional	Location of a file that provides information for consumers about the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
consumer-documentation-type	String	Optional	Type of the consumer documentation file, either text or pdf. This is required if consumer-documentation-file is specified with a value that is not null.
admin-documentation-file	String	Optional	Location of a file that provides information for administrators about the template. Specify the fully qualified z/OS UNIX path of the file, beginning with the forward slash (/) and including the file name.
admin-documentation-type	String	Optional	Type of the administrator documentation file, either text or pdf. This is required if admin-documentation-file is specified with a value that is not null.
approvals	Array of strings	Optional	General approvals that are associated with the template. Each string represents the user ID of a general approval. If the array contains a user ID for a general approval that already exists for the template, the status and all of the corresponding information for that user ID is maintained for the template. If the array does not contain a user ID for a general approval that already exists for the template, that user ID is removed from the general approval list for the template. An empty array removes any existing general approvals from the template. A null value for approvals results in no changes.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Normal) is returned.

Example HTTP interaction

Figure 39 shows a request to modify a software services template.

```
{
"action-definition-file": "/Users/gg/zosmf/user-workflows/factory/mq/qmgr/qmgrActionsUpdate.xml"
}
```

Figure 39. Sample request to modify a software services template

Delete a software services template

You can use this operation to delete a software services template from the catalog.

HTTP method and URI path

```
DELETE /zosmf/provisioning/rest/<version>/scc/<object-id>
```

In this request

<object-id>
Identifies the software services template to be deleted.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation deletes a software services template from the catalog.

On successful completion, HTTP status code `204` (Normal) is returned, indicating that the request resulted in a software services template being deleted.

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Normal) is returned.

Example HTTP interaction

Figure 40 shows a request to delete a software services template.

```
DELETE https://localhost:4444/zosmf/provisioning/rest/1.0/scc/b69c4212-c7c9-422b-8835-de185e16bb23
```

Figure 40. Sample request to delete a software services template

Get a software services template

Use this operation to retrieve a software services template from the catalog.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template to retrieve.

Query parameters

None.

Description

This operation retrieves a software services template from the catalog.

On successful completion, the operation returns HTTP status code `200` (OK), indicating that the request resulted in a software services template being retrieved. A response body is provided, as described in "Response content"

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services template. Table 67 on page 99 lists the fields in the JSON object.

Table 67. Response from a get software services template request

Field	Type	Description
base-object-id	String	The object ID that is associated with all of the versions of the software services template.
generated-name	String	Generated name for the software services template.
name	String	The name associated with the software services template.
version	String	Version of the software services template.
owner	String	User ID of the software services template owner.
state	String	Indicates the status of the software services template. See "State values" on page 100
description	String	Description of the software services template.
tenants	Array of Strings	Each string represents a tenant that the template is associated with.
domain-name	String	The domain the template is associated with.
approvals	Array of objects	Array of Approval-Object containing information about the approvals associated with this software services template. See Table 68 on page 101.
action-definition-file	String	Location of the action definition file.
action-definition-file-original-source	String	Original user specified location of the action definition file
action-definition-file-original-timestamp	String	Last-modified time stamp for when the original action definition file source was specified, in ISO 8601 format.
actions	Array of objects	Array of Action-Object containing information about the actions associated with the template. See Table 69 on page 101.
software-id	String	A short, arbitrary, value that identifies the software that is being provisioned.
software-name	String	Name of the software that is being provisioned.
software-type	String	Type of software that is being provisioned.
software-version	String	Version of the software that is being provisioned.
workflow-definition-file	String	Location of the workflow definition file, the primary XML file that defines the workflow
workflow-definition-file-original-source	String	Original user-specified location of the workflow definition file.
workflow-definition-file-original-timestamp	String	The last-modified time stamp for when the original workflow definition file source was specified, in ISO 8601 format.
workflow-id	String	A short, arbitrary value that identifies the workflow.
workflow-vendor	String	Name of the vendor that provided the workflow definition file.
workflow-version	String	Version of the workflow definition file.
workflow-variable-input-file	String	Location of the workflow variable input file, an optional properties file used to specify in advance the values for one or more of the variables that are defined in the workflow definition file.
workflow-variable-input-file-original-source	String	The original user-specified location of the workflow variable input file.
workflow-variable-input-file-original-timestamp	String	The last-modified time stamp for when the original variable input file source was specified, in ISO 8601 format.

Table 67. Response from a get software services template request (continued)

Field	Type	Description
workflow-clean-after-provisioned	Boolean	Indicates if a workflow that performs provisioning should be automatically deleted after it completes successfully. The value is true to delete, false to keep. The default value, if none is specified, is false, which keeps the workflow.
prompt-variables	Array of objects	Array of prompt variable objects containing information about the variables that are expected to be prompted for in preparation for running the software services template. See Table 71 on page 102.
at-create-variables	Array of strings	Array of strings that name the variables that are either prompt variables (variables that are expected to be prompted for in preparation for running the software services template), or required variables (variables for which a value is required when the software services template is run), or both.
consumer-documentation-file	String	Location of the original file that provides information for consumers about the template.
consumer-documentation-type	String	Type of the consumer documentation file, either text or pdf. This is required if consumer-documentation-file is specified.
admin-documentation-file	String	Location of a file that provides information for administrators about the template.
admin-documentation-type	String	Type of the administrator documentation file, either text or pdf. This is required if admin-documentation-file is specified.
create-time	String	Time that this object was created, in ISO 8601 format.
create-by-user	String	User who created this object.
last-modified-time	String	The last time this object was updated, in ISO 8601 format.
last-modified-by-user	String	User who last updated this object.

State values

archived
The entry is hidden from consumers. You can make it available again with the **Publish** action.

corrupted
The contents of the software services template are missing or incorrect. Delete the template.

draft The entry is in the edit state and visible only to the owner and the administrator. No approvals are required for this template. The entry can be tested with the Test a software services template API.

draft_approved
The software services template is in edit state and all the approvals that are associated with the template and the respective runAs user IDs have been received. The entry can be tested with the Test a software services template API.

draft_pending_approvals
The software services template is in edit state and one or more associated approvals has not been approved. The entry cannot be tested (with the Test a software services template API) in this state.

draft_missing_required_approver
One or more of the definition files contains a runAsUser element without a corresponding approver element. Either an approver element must be added for the runAsUser element, or a domain or general approver must be added for the software services template. The entry cannot be tested (with the Test a software services template API) in this state.

pending_security_update
Permission to access the software services template is being processed. No API requests are allowed for the software services template until the security processing is complete.

published
The entry is locked and visible to consumers.

security_update_failed
Security access setup related to the software services template failed. Only the view and delete API requests a available.

Table 68. Response from a get request: Approval-Object

Field	Type	Description
status	String	Status of the approval for this object: pending, approved, or rejected.
comment	String	Comment associated with the change in status from pending to either approved or rejected.
description	String	Additional detail that is provided if the approval is for a workflow definition that is associated with the action definition, for example, This workflow definition is associated with the <action-name> action.
user-ids	Array of strings	Each string in the array is a user ID that can approve the template, workflow step, or action. Any one of the user IDs in the array can approve or reject. The last action takes precedence.
status-update-by	String	User ID that performed the last approve or reject action for this approval object.
time-of-update	String	The last time this object was updated, in ISO 8601 format.
run-as-user	String	The runAsUser user ID that the approval object is for. Only applicable when the type is action_definition or step_definition.
type	String	Type of approval object: general, domain, action_definition, or step_definition.
object-id	String	Unique object ID representing this approval object.
workflow-file	String	Workflow file definition associated with this runAsUser user ID.
variable-input-file	String	Variable input file associated with this runAsUser user ID.
step-name	String	Workflow file definition step associated with this runAsUser user ID.
called-by-step-name	String	Step in the parent workflow definition that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
called-by-workflow-file	String	Workflow definition that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
actions-file	String	Actions file definition associated with this runAsUser user ID.
action-name	String	Action that is defined in the actions file associated with this runAsUser user ID.

Table 69. Response from a get request: Action-Object

Field	Type
name	String
type	String
is-deprovision	String. The value must be either true or false.

Table 69. Response from a get request: Action-Object (continued)

Field	Type
command	String
command-run-as-user	String
command-sol-key	String
command-unsol-key	String
command-detect-time	String
workflow-definition-file	String
workflow-variable-input-file	String
workflow-variables	Variable[]
instructions	String
prompt-variables	The prompt variable objects that are associated with the action.

Table 70. Response from a get request: Variable-Object

Field	Type
name	String
value	String
visibility: public or private	String

Table 71. Response from a get request: Prompt-Variable-Object

Field	Type	Description
name	String	Name of the property.
value	String	Current value for the property.
required	boolean	Indicates whether the variable value is required during the workflow create process.
label	String	Short label for the UI widget.
description	String	Explanation of what the variable is used for and perhaps what the syntactic requirements are.
abstract	String	Brief description of the variable for the UI widget.
type	String	Type of the variable element: boolean, string, integer, decimal, time, date.
must-be-choice	boolean	Indicates whether the value must come from the provided choices.
choices	Array of Strings	Contains allowable choices for the value of the variable.
regex	String	Standard regular expression that constrains the variable value.
multi-line	boolean	Indicates whether the value requires a multi-line text box.
min	String	For a string type, indicates the minimum string length of the value. For all other types, indicates the minimum value required.
max	String	For a string type, indicates the maximum string length of the value. For all other types, indicates the maximum value required.
places	String	Maximum number of decimal places that can be specified for a variable of type decimal.
error-message	String	Default error message associated with an incorrect value.

Fields of type String default to null.

Example HTTP interaction

In Figure 41, a request is submitted to retrieve a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/5ccbad22-94fd-4b31-bb2b-95aa8602cc48
```

Figure 41. Sample request to retrieve a software services template

The following is the response body for the example GET request.


```
{
  "name": "mqCBA",
  "version": "1",
  "owner": "domadmin",
  "state": "published",
  "description": "This workflow provisions an MQ for z/OS Queue Manager",
  "tenants": [...],
  "actions": [...],
  "approvals": [],
  "tested": false,
  "generated-name": "mqCBA.1.default",
  "domain-name": "default",
  "action-definition-file": "definition/qmgrActions.xml",
  "action-definition-file-original-source": "/users/gg/mqCBA/definition/qmgrActions.xml",
  "action-definition-file-original-timestamp": "2016-11-18T20:00:42Z",
  "software-id": "5655-W97",
  "software-name": "IBM MQ for z/OS",
  "software-type": "QMgr",
  "software-version": "V8.0.0",
  "workflow-definition-file": "definition/provision.xml",
  "workflow-definition-file-original-source": "/users/gg/mqCBA/definition/provision.xml",
  "workflow-definition-file-original-timestamp": "2016-11-18T20:03:47Z",
  "workflow-id": "ProvisionQueueManager",
  "workflow-vendor": "IBM",
  "workflow-version": "1.0.1",
  "workflow-variable-input-file": "definition/workflow_variables.properties",
  "workflow-variable-input-file-original-source": "/users/gg/mqCBA/definition/workflow_variables.properties",
  "workflow-variable-input-file-original-timestamp": "2016-11-18T20:00:42Z",
  "prompt-variables": [],
  "at-create-variables": [],
  "workflow-clean-after-provisioned": true,
  "security-wf-info": null,
  "create-time": "2016-11-18T20:00:43.504Z",
  "created-by-user": "domadmin",
  "last-modified-by-user": "domadmin",
  "last-modified-time": "2016-11-18T20:04:50.913Z",
  "admin-documentation-file-original-source": "/users/gg/mqCBA/documentation/admin-mqaas_readme.pdf",
  "admin-documentation":
     "/zosmf/provisioning/rest/1.0/scc/5b0c3367-b856-4727-99ac-f9a79c9abf28/documentation/admin",
  "admin-documentation-type": "pdf",
  "consumer-documentation-file-original-source":
     "/users/gg/mqCBA/documentation/consumer-workflow_variables.properties",
  "consumer-documentation":
     "/zosmf/provisioning/rest/1.0/scc/5b0c3367-b856-4727-99ac-f9a79c9abf28/documentation/consumer",
  "consumer-documentation-type": "text",
  "base-object-id": "c0e4d08f-f046-4a79-8a15-6981743d07ed",
  "admin-documentation-mime-type": "application/pdf",
  "consumer-documentation-mime-type": "text/plain"
}
```

Get software services template documentation

Use this operation to retrieve software services template documentation from the catalog.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/documentation/admin
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/documentation/consumer
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template to retrieve.

documentation/admin
Causes the administrator documentation file to be retrieved.

documentation/consumer
Causes the consumer documentation file to be retrieved.

Query parameters

None.

Description

This operation retrieves software services template documentation from the catalog.

On successful completion, the operation returns HTTP status code `200` (OK), indicating that the request resulted in software services template documentation being retrieved.

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

The documentation file in the associated mime type.

Example HTTP interaction

In Figure 42, a request is submitted to retrieve the consumer documentation for a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/5ccbad22-94fd-4b31-bb2b-95aa8602cc48/documentation/consumer
```

Figure 42. Sample request to retrieve software services template documentation

Get prompt variables for a software services template

Use this operation to retrieve the variables that are required to run the software services template and for which a prompt can be used to obtain the value.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/prompt-variables
```

In this request

<object-id>
Identifies the software services template to be retrieved.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation retrieves the variables for which a prompt can obtain the value.

On successful completion, HTTP status code `200` (Normal) is returned, indicating that the request resulted in a software services template being retrieved. A response body is provided, as described in "Response content."

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the prompt variables. See Table 72 on page 107 and Table 73 on page 107.

Table 72. Response from a get prompt variables request

Field	Type	Required/ Optional	Description
prompt-variables	Array of objects	Required	Array of Prompt-Variable-Object containing information about the variables prompted at create time. See .

Table 73. Response from a get request: Prompt-Variable-Object

Field	Type	Description
name	String	Name of the property.
value	String	Current value for the property.
required	boolean	Indicates whether the variable value is required during the workflow create process.
label	String	Short label for the UI widget.
description	String	Explanation of what the variable is used for and perhaps what the syntactic requirements are.
abstract	String	Brief description of the variable for the UI widget.
type	String	Type of the variable element: boolean, string, integer, decimal, time, date.
must-be-choice	boolean	Indicates whether the value must come from the provided choices.
choices	Array of Strings	Contains allowable choices for the value of the variable.
regex	String	Standard regular expression that constrains the variable value.
multi-line	boolean	Indicates whether the value requires a multi-line text box.
min	String	For a string type, indicates the minimum string length of the value. For all other types, indicates the minimum value required.
max	String	For a string type, indicates the maximum string length of the value. For all other types, indicates the maximum value required.
places	String	Maximum number of decimal places that can be specified for a variable of type decimal.
error-message	String	Default error message associated with an incorrect value.

Fields of type String default to null.

Example HTTP interaction

Figure 43 shows a request to retrieve the prompt variables.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/7953ecc0-3b28-4ef4-a72b-18ba854d10d3/prompt-variables
```

Figure 43. Sample request to retrieve prompt variables

The following is the response body for the request.

```
{
   "prompt-variables": [
     {
       "name": "CSQ_MQ_SSID",
       "label": "MQ_SSID",
       "description": "The name of the MQ subsystem to be provisioned.",
       "type": "string",
       "value": "ZCT1",
       "required": true,
       "choices": null,
       "regex": "[A-Z0-9]{1,4}",
       "min": null,
       "max": null,
       "places": null,
       "abstract": "Subsystem identifier",
       "multi-line": false,
       "must-be-choice": false,
       "error-message": "The value entered is not valid."
     },
     {
       "name": "CSQ_ENVIRONMENT",
       "label": "ENVIRONMENT",
       "description": "The environment for which the queue manager is to be provisioned/de-provisioned.
        The BSDS, Log and Pageset datasets vary depending on the environment.",
       "type": "string",
       "value": "TEST",
       "required": true,
       "choices": [
         "DEV",
         "TEST",
         "QA",
         "PROD"
       ],
       "regex": null,
       "min": null,
       "max": null,
       "places": null,
       "abstract": "Environment for which the queue manager is to be provisioned/de-provisioned
       (DEV, TEST, QA, PROD)",
       "multi-line": false,
       "must-be-choice": true,
       "error-message": "The value entered is not valid."
     }
   ]
 }
```

Figure 44. Response body for the GET prompt variables request

Get source information for a software services template

Use this operation to retrieve source information for a software services template .

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/sources
```

In this request

<object-id>
Identifies the software services template for which information is to be retrieved.

<version>
Is the URI path variable <*version*> that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation retrieves source information for a template, which includes details of the original source paths that were provided, and whether the files have been changed since the last time the template was updated with the source paths.

On successful completion, HTTP status code `200` (Normal) is returned, indicating that the request resulted in a software services template being retrieved. A response body is provided, as described in "Response content."

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object. See Table 74 on page 110.

Table 74. Response from a get source request

Field	Type	Description
action-definition-file	Source-Info-Object	Details for the action definition file. See Table 75.
workflow-definition-file	Source-Info-Object	Details for the workflow definition file. See Table 75.
workflow-variable-input-file	Source-Info-Object	Details for the workflow variable input file. See Table 75.

Table 75. Response from a get request: Source-Info-Object

Field	Type	Description
original-source-path	String	The original source path provided for the file.
out-of-sync	boolean	Indicates if the file that is associated with the template matches the original source file. The value is false if the current file that is associated with the template matches the original source file, and true if the current file that is associated with the template differs from the original source file, or if the original source file is not found.

Example HTTP interaction

Figure 45 shows a request to retrieve the source information.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/5ccbad22-94fd-4b31-bb2b-95aa8602cc48/sources
```

Figure 45. Sample request to retrieve source information

The following is the response body for the request.

```
{
  "action-definition-file": {
    "original-source-path": "/users/gg/zosmf/user-workflows/factory/mq/qmgr/qmgrActions.xml",
    "out-of-sync": false
  },
  "workflow-definition-file": {
    "original-source-path": "/users/gg/zosmf/user-workflows/factory/mq/qmgr/provision.xml",
    "out-of-sync": false
  },
  "workflow-variable-input-file": null
}
```

Figure 46. Response body for the get source request

List the software services templates

You can use this operation to list the software services templates that are defined in the catalog.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

You can specify the following query parameter on this request. Objects matching all query parameters are returned.

name
Optional, regular expression, specifies the external name of the software services template.

owner
Optional, specifies the user ID or group ID that identifies the owner of the software services template.

software-type
Optional, specifies the type of software being provisioned.

domain-name
Optional, specifies the domain name.

state
Optional, regular expression, specifies the state.

If you specify no query parameters, then all software services templates are returned.

Description

This operation list the software services templates in the catalog.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a list of software services templates being retrieved. A response body is provided, as described in "Response content" on page 112.

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services templates. See Table 76, Table 77.

Table 76. Array of objects

Field	Type	Description
scc-list	Array of objects	Array of software services template objects. The array is filter based on any query parameters that were provided.

Table 77. Fields for each software services template

Field	Type	Description
generated-name	String	The generated name for the software services template.
object-id	String	The unique ID that identifies the software services template.
base-object-id	String	The object ID that is associated with all of the versions of the software services template.
name	String	Descriptive name for the software services template.
version	String	Version of the software services template.
owner	String	User ID of the software services template owner.

Table 77. Fields for each software services template (continued)

Field	Type	Description
state	String	Indicates the status of the software services template: **archived** The entry is hidden from consumers. You can make it available again with the **Publish** action. **corrupted** The contents of the software services template are missing or incorrect. Delete the template. **draft** The entry is in the edit state and visible only to the owner and the administrator. No approvals are required for this template. The entry can be tested with the Test a software services template API. **draft_approved** The software services template is in edit state and all the approvals that are associated with the template and the respective runAs user IDs have been received. The entry can be tested with the Test a software services template API. **draft_pending_approvals** The software services template is in edit state and one or more associated approvals has not been approved. The entry cannot be tested (with the Test a software services template API) in this state. **draft_missing_required_approver** One or more of the definition files contains a runAsUser element without a corresponding approver element. Either an approver element must be added for the runAsUser element, or a domain or general approver must be added for the software services template. The entry cannot be tested (with the Test a software services template API) in this state. **pending_security_update** Permission to access the software services template is being processed. No API requests are allowed for the software services template until the security processing is complete. **published** The entry is locked and visible to consumers. **security_update_failed** Security access setup related to the software services template failed. Only the view and delete API requests a available.
description	String	Description of the software services template.
action-definition-file	String	Location of the action definition file, a file in XML format that defines the actions for the software services instance provisioned from the template.
software-id	String	A short, arbitrary value that identifies the software that is being provisioned.
software-name	String	Name of the software that is being provisioned.
software-type	String	Identifies the type of software that is being provisioned.
software-version	String	Version of the software that is being provisioned.
workflow-definition-file	String	Location of the workflow definition file, the primary XML file that defines the workflow.
workflow-id	String	Workflow ID. A short, arbitrary value that identifies the workflow.

Table 77. Fields for each software services template (continued)

Field	Type	Description
workflow-vendor	String	Name of the vendor that provided the workflow definition file.
workflow-version	String	Version of the workflow definition file.
workflow-variable-input-file	String	Location of the workflow variable input file, an optional properties file used to specify in advance the values for one or more of the variables that are defined in the workflow definition file.
domain-name	String	The name of the domain that the template resides in.
create-time	String	The time that this object was created, in ISO 8601 format.
created-by-user	String	The user that created this object.
last-modified-time	String	The last time this object was updated, in ISO 8601 format.
last-modified-by-user	String	The user that last updated this object.

Example HTTP interaction

Figure 47 shows a request to retrieve a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/
```

Figure 47. Sample request to retrieve a software services template

The following is the response body for the request.


```
{
  "scc-list": [
    {
      "name": "mqCBA",
      "version": "1",
      "owner": "domadmin",
      "state": "published",
      "description": "This workflow provisions an MQ for z/OS Queue Manager",
      "generated-name": "mqCBA.1.default",
      "object-id": "5b0c3367-b856-4727-99ac-f9a79c9abf28",
      "base-object-id": "c0e4d08f-f046-4a79-8a15-6981743d07ed",
      "domain-name": "default",
      "action-definition-file": "definition/qmgrActions.xml",
      "software-id": "5655-W97",
      "software-name": "IBM MQ for z/OS",
      "software-type": "QMgr",
      "software-version": "V8.0.0",
      "workflow-definition-file": "definition/provision.xml",
      "workflow-id": "ProvisionQueueManager",
      "workflow-vendor": "IBM",
      "workflow-version": "1.0.1",
      "workflow-variable-input-file": "definition/workflow_variables.properties",
      "create-time": "2016-11-18T20:00:43.504Z",
      "created-by-user": "domadmin",
      "last-modified-by-user": "domadmin",
      "last-modified-time": "2016-11-18T20:28:43.951Z"
    }
  ]
}
```

Publish a software services template

You can use this operation to publish a software services template. The publish operation locks the template, preventing any further modification, and creates a public copy of it.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/publish
```

In this request

<object-id>
Identifies the software services template to be published.

<version>
Is the URI path variable <*version*> that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation publishes a software services template.

On successful completion, HTTP status code `204` (Normal) is returned, indicating that the request resulted in a software services template being published.

The software services template must be in the draft, draft approved, or archived state.

To work with a published software services template, use the REST APIs that are described in "Published software service template services" on page 138.

Request content

The request body is optional. It contains a JSON object that describes the publish operation. See Table 78.

Table 78. Request content for the software services template request

Parameter	Type	Required or Optional	Description
archive-existing	boolean	Optional	If set to true, indicates that if a published entry with this name already exists, that entry should be moved into the archived state, and publish this one instead. If set to false, indicates that if a published entry with this name already exists, the request should fail. False is the default if this parameter is not specified. If no published entry with this name already exists, then this flag is ignored.

Table 78. Request content for the software services template request (continued)

Parameter	Type	Required or Optional	Description
ignore-test	boolean	Optional	If set to true, indicates a publish of the template does not require a test run to be performed. If this parameter is not specified, then the value defaults to false.
ignore-source-change	boolean	Optional	If set to true, indicates that the publish of the entry is not restricted by the change in the original source that was used on the create or modify of the entry. If this parameter is not specified, then the value defaults to false.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Normal) is returned.

Response content

None.

Example HTTP interaction

Figure 48 shows a request to publish a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/d0166782-4e18-4b07-a075-c8946c88e068/actions/publish
```

Figure 48. Sample request to publish a software services template

Test a software services template

You can use this operation to test a software services template.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/test
```

In this request

<object-id>
Identifies the software services template to be published.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation lets you perform a test run of a software services template. The test run creates a workflow entry, starts the workflow entry, and creates a software services registry entry in being provisioned state.

On successful completion, HTTP status code `200` (Normal) is returned, indicating that the entry in the registry was created.

All approvals for a software services template must be approved before it can be tested. The software services template must be in the draft or draft approved state.

The software services template must be in the draft or draft_approved state.

Request content

The request content is expected to contain a JSON object that describes the test run. See Request content for the software services template request and Table 80 on page 118.

Table 79. Request content for the test software services template request

Field name	Type	Required or optional	Description
input-variables	Array of objects	Optional	Array of required run-time property objects.
user-data-id	String	Optional	ID of user-data. The user-data-id and user-data values are associated with the software services instance that is created and are returned with requests for the software services instance.
user-data	String	Optional	User-supplied data to be associated with the software services instance. Only allowed if user-data-id is also provided.
tenant-name	String	Optional	Required if the template is associated with more than one tenant

Table 79. Request content for the test software services template request (continued)

Field name	Type	Required or optional	Description
account-info	String	Optional	Account information to use in the JCL JOB statement. By default, it is the account information that is associated with the tenant resource pool

Table 80. Runtime properties

Field	Type	Description
name	String	Name of the runtime property.
value	String	Value of the runtime property.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (Normal) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services template request. Table 79 on page 117 lists the fields in the JSON object.

Table 81. Response from a test software services template request

Field	Type	Description
registry-info	Object	Object mapping matching the return response body from a registry create.
workflow-info	Object	Object mapping matching the return response body from a workflow create.

Example HTTP interaction

Figure 49 shows a request to test a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/f2cd93d7-3b13-4fa0-9fb7-d36a93c358e8/actions/test

{
     "input-variables":[{"name":"CSQ_MQ_SSID","value":"ZCT1"},
      {"name":"CSQ_CMD_PFX","value":"!ZCT1"},{"name":"CSQ_ENVIRONMENT","value":"TEST"}],
     "tenant-name":"tenant1",
     "user-data-id":"10252015",
     "user-data":"scout is an amazing welshie puppy"
}
```

Figure 49. Sample request to test a software services template

The following is the response body for the request.

```
{
   "registry-info": {
     "object-name": "QMgr_5",
     "object-id": "1d5fbad5-11e2-4bdf-bd81-02c0e322f71f",
     "object-uri": "/zosmf/provisioning/rest/1.0/scr/1d5fbad5-11e2-4bdf-bd81-02c0e322f71f"
   },
   "workflow-info": {
     "workflowKey": "4b917e88-d2db-4819-a7a1-6403a99f65d4",
     "workflowDescription": "Procedure to provision a MQ for zOS Queue Manager",
     "workflowID": "ProvisionQueueManager",
     "workflowVersion": "1.0.1",
     "vendor": "IBM"
   }
 }
```

Figure 50. Sample response body

Refresh a software services template

You can use this operation to refresh the files that are associated with a software services template.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/refresh
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template that the approval is associated with.

<approval-object-id>
Identifies the approval object to delete.

Query parameters

None.

Description

This operation obtains the latest contents of the files that are associated with a software services template: the workflow definition XML file, the actions XML file, and the workflow variable input file if one is specified, and any documentation files that are provided. The information in the software services template is updated to reflect the latest contents, including timestamps, of those files. The files are located by the original source paths.

Refresh causes all approvals to be reset.

On successful completion, HTTP status code `204` (Successful) is returned.

The software services template must be in one of the draft states.

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Successful) is returned.

Example HTTP interaction

In Figure 51, a request is submitted to refresh a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/d0166782-4e18-4b07-a075-c8946c88e068/actions/refresh
```

Figure 51. Sample request to refresh a software services template

Archive a software services template

You can use this operation to archive a published software services template.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/actions/archive
```

In this request

<object-id>
Identifies the software services template to be archived.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation lets you archive a software services template. This puts the software services template in an archived state.

The software services template must be in a published state.

On successful completion, HTTP status code `204` (Normal) is returned, indicating that the archive was successful.

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord and a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Normal) is returned.

Response content

None.

Example HTTP interaction

In Figure 52 on page 123, a request is submitted to archive a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/1234-1223-23232/actions/archive
```

Figure 52. Sample request to archive a software services template

The following is the response body for the request.

```
204 No Content
```

Figure 53. Sample response body

Add an approval for a software services template

You can use this operation to create an approval record for a software services template. The approval record associates a user ID with the software services template.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals
```

In this request

<object-id>
Identifies the software services template that the approval is associated with.

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation lets you create a new general approval record for a software services template. It returns a unique approval object ID that identifies the approval. The approval is for all of the contents of the software services template and is not associated with a specific step or action.

The software services template must be in one of the draft states.

All approvals for a software services template must be approved before it can be published or tested. Once all the approvals are approved the state of the entry is updated to draft_approved.

On successful completion, HTTP status code `201` (Normal) is returned, indicating that the approval was created.

Request content

The request content is expected to contain a JSON object that describes the approval record. See Request content for the software services template request.

Table 82. Request content for the add approval request

Field name	Type	Required or optional	Description
user-id	String	Required	User ID associated with this approval.

Authorization requirements

The user's z/OS user ID must be defined as a landlord or a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `201` (Normal) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services template request. Table 82 on page 124 lists the fields in the JSON object.

Table 83. Response from an add approval request

Field	Type	Description
object-id	String	Object ID of the newly created approval. The object ID is to be used in subsequent requests to the session.
object-uri	String	URI of the newly created approval.

Example HTTP interaction

In Figure 54, a request is submitted to add an approval record for a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/approvals

{
    "user-id":"nick"
}
```

Figure 54. Sample request to add an approval record for a software services template

The following is the response body for the request.

```
{
    "object-id": "eeb4f5a3-d883-4190-9961-412306707426",
    "object-uri": "/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/
     approvals/eeb4f5a3-d883-4190-9961-412306707426"
}
```

Figure 55. Sample response body

Get an approval for a software services template

You can use this operation to retrieve an approval for a software services template.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/
<approval-object-id>
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template that the approval is associated with..

<approval-object-id>
Identifies the approval to retrieve.

Query parameters

None.

Description

This operation retrieves an approval for a software services template.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in an approval being retrieved. A response body is provided, as described in "Response content"

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the approval. Table 84 on page 127 lists the fields in the JSON object.

Table 84. Response from a get approval request

Field	Type	Description
status	String	Status of the approval for this object: pending, approved, or rejected.
comment	String	Comment that is associated with the change in status from pending to either approved or rejected.
description	String	Additional detail that is provided if the approval is for a workflow definition that is associated with the action definition.
user-ids	Array of strings	Each string in the array is a user ID that can approve the template, workflow step, or action. Any one of the user IDs in the array can approve or reject. The last action takes precedence.
status-update-by	String	User ID that performed the last approve or reject action for this approval object.
time-of-update	String	The last time this object was updated, in ISO 8601 format.
run-as-user	String	The runAsUser user ID that the approval object is for. This applies only to action_definition and step_definition types.
type	String	Type of approval object: general, domain, action_definition, or step_definition.
object-id	String	Unique object ID representing this approval object.
workflow-file	String	Workflow file definition that is associated with this runAsUser user ID.
variable-input-file	String	Specifies the variable input file that is associated with this runAsUser user ID.
step-name	String	Workflow file definition step that is associated with this runAsUser user ID.
called-by-step-name	String	Step in the parent workflow definition that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
called-by-workflow-file	String	Workflow definition file that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
actions-file	String	Actions definition file that is associated with this runAsUser user ID.
action-name	String	Action defined in the actions definition file that is associated with this runAsUser user ID.

Example HTTP interaction

In Figure 56, a request is submitted to get an approval record for a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/
approvals/dacea656-ffbe-48ce-a193-575161ff9d43
```

Figure 56. Sample request to get an approval record for a software services template

The following is the response body for the request.

```
{
  "status": "approved",
  "comment": "approved",
  "description": "General approval for all of the template contents.",
  "type": "general",
  "object-id": "dacea656-ffbe-48ce-a193-575161ff9d43",
  "user-ids": [
    "zosmfad"
  ],
  "status-update-by": "zosmfad",
  "time-of-update": "2016-11-18T19:13:00.435Z",
  "run-as-user": null,
  "workflow-file": null,
  "variable-input-file": null,
  "step-name": null,
  "called-by-step-name": null,
  "called-by-workflow-file": null,
  "actions-file": null,
  "action-name": null
}
```

Figure 57. Sample response body

List the approvals for a software services template

You can use this operation to list all of the approvals for a software services template.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template that the approval is associated with.

Query parameters

None.

Description

This operation retrieves all of the approval for a software services template.

On successful completion, HTTP status code `200` (Normal) is returned. A response body is provided, as described in "Response content"

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord, domain administrator, domain approver, or template approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (Normal) is returned.

Response content

On successful completion, the service returns a JSON oject. See Table 85 and Table 86 on page 130.

Table 85. Response from a list approvals request

Field	Type	Description
approvals	Array of objects	Array of Approval-Object containing information about the approvals associated with this software services template.

Table 86. Response from a get approval request

Field	Type	Description
status	String	Status of the approval for this object: pending, approved, or rejected.
comment	String	Comment that is associated with the change in status from pending to either approved or rejected.
description	String	Additional detail that is provided if the approval is for a workflow definition that is associated with the action definition.
user-ids	Array of strings	Each string in the array is a user ID that can approve the template, workflow step, or action. Any one of the user IDs in the array can approve or reject. The last action takes precedence.
status-update-by	String	User ID that performed the last approve or reject action for this approval object.
time-of-update	String	The last time this object was updated, in ISO 8601 format.
run-as-user	String	The runAsUser user ID that the approval object is for. This applies only to action_definition and step_definition types.
type	String	Type of approval object: general, domain, action_definition, or step_definition.
object-id	String	Unique object ID representing this approval object.
workflow-file	String	Workflow file definition that is associated with this runAsUser user ID.
variable-input-file	String	Specifies the variable input file that is associated with this runAsUser user ID.
step-name	String	Workflow file definition step that is associated with this runAsUser user ID.
called-by-step-name	String	Step in the parent workflow definition that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
called-by-workflow-file	String	Workflow definition file that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
actions-file	String	Actions definition file that is associated with this runAsUser user ID.
action-name	String	Action defined in the actions definition file that is associated with this runAsUser user ID.

Example HTTP interaction

In Figure 58, a request is submitted to get an approval record for a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/approvals
```

Figure 58. Sample request to list the approval records for a software services template

The following is the response body for the request.

```
{
  "approvals": [
    {
      "status": "pending",
      "comment": null,
      "description": "The approver element originates from the 'qmCriteria' step in the
        /users/gg/mqCBA/definition/provision.xml which is the primary workflow definition file.",
      "type": "step_definition",
      "object-id": "70c54ad6-11ad-41ea-9682-670fa429438b",
      "user-ids": [
        "scout",
        "nick"
      ],
      "status-update-by": null,
      "time-of-update": null,
      "run-as-user": "zosmfad",
      "workflow-file": "/users/gg/mqCBA/definition/provision.xml",
      "variable-input-file": "/Users/gg/mqCBA/definition/workflow_variables.properties",
      "step-name": "qmCriteria",
      "called-by-step-name": null,
      "called-by-workflow-file": null,
      "actions-file": null,
      "action-name": null
    },
    {
      "status": "pending",
      "comment": null,
      "description": "General approval for all of the template contents.",
      "type": "general",
      "object-id": "76bc1762-c003-4bb1-9b0e-df28ef4f986d",
      "user-ids": [
        "zosmfad"
      ],
      "status-update-by": null,
      "time-of-update": null,
      "run-as-user": null,
      "workflow-file": null,
      "variable-input-file": null,
      "step-name": null,
      "called-by-step-name": null,
      "called-by-workflow-file": null,
      "actions-file": null,
      "action-name": null
    }
  ]
}
```

Figure 59. Sample response body

Approve an approval record for a software services template

You can use this operation to approve the contents of approval record for a software services template.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/
<approval-object-id>/actions/approve
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template that the approval is associated with.

<approval-object-id>
Identifies the approval object to approve.

Query parameters

None.

Description

This operation approves the contents of an approval record for the software services template.

On successful completion, HTTP status code `204` (Successful) is returned.

The software services template must be in one of the draft states.

Request content

The request content is expected to contain a JSON object that describes the approval. See Table 87.

Table 87. Request content for the software services template request

Field name	Type	Required or optional	Description
comment	String	Optional	Text describing the approval.

Authorization requirements

The user's z/OS user ID must be defined as a domain approver or template approver, or be one of the approvers in the approval object.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Successful) is returned.

Example HTTP interaction

In Figure 60, a request is submitted to approve an approval record for a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/approvals/
dacea656-ffbe-48ce-a193-575161ff9d43/actions/approve
```

Figure 60. Sample request to approve an approval record for a software services template

Reject the use of your user ID with a software services template

You can use this operation to reject the use of your user ID with a software services template.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/
<approval-object-id>/actions/reject
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template that the approval is associated with.

<approval-object-id>
Identifies the approval object to reject.

Query parameters

None.

Description

This operation rejects an approval that is associated with a software services template. Rejecting the approval means that your user ID is not allowed to be used with the software services template.

On successful completion, HTTP status code `204` (Successful) is returned.

The software services template must be in one of the draft states.

Request content

The request content is expected to contain a JSON object that describes the rejection. See Table 88.

Table 88. Request content for the software services template request

Field name	Type	Required or optional	Description
comment	String	Optional	Text describing the rejection.

Authorization requirements

The user's z/OS user ID must be defined as a domain approver or template approver, or be one of the approvers in the approval object.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Successful) is returned.

Example HTTP interaction

In Figure 61, a request is submitted to reject an approval record for a software services template.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/approvals/
dacea656-ffbe-48ce-a193-575161ff9d43/actions/reject

{
     "comment":"disagree with this, rework required"
}
```

Figure 61. Sample request to reject an approval record for a software services template

Delete an approval for a software services template

You can use this operation to delete an approval that is associated with a software services template.

HTTP method and URI path

```
DELETE /zosmf/provisioning/rest/<version>/scc/<object-id>/approvals/
<approval-object-id>
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<object-id>
Identifies the software services template that the approval is associated with.

<approval-object-id>
Identifies the approval object to delete.

Query parameters

None.

Description

This operation deletes an approval that is associated with a software services template.

On successful completion, HTTP status code `204` (Successful) is returned.

The software services template must be in one of the draft states.

Request content

None.

Authorization requirements

The user's z/OS user ID must be defined as a landlord or a domain administrator.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `204` (Successful) is returned.

Example HTTP interaction

In Figure 62 on page 137, a request is submitted to delete an approval record for a software services template.

```
DELETE https://localhost:4444/zosmf/provisioning/rest/1.0/scc/8abd70b5-ac74-4f4a-bc09-266bf7cf8270/approvals/
dacea656-ffbe-48ce-a193-575161ff9d43
```

Figure 62. Sample request to delete an approval record for a software services template

Published software service template services

The published software service template services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to create and manage software services templates that are in the published state.

For information about cloud provisioning, including a description of the roles, see "Cloud provisioning services" on page 46.

The published software services catalog contains a list of the software services templates that are in the published state.

Table 89 lists the operations that the published software service template services provide.

Table 89. z/OSMF published software service template services: operations summary

Operation name	HTTP method and URI path
"Run a published software service template" on page 140	`POST /zosmf/provisioning/rest/<version>/psc/<name>/actions/run`
"Get a published software service template" on page 143	`GET /zosmf/provisioning/rest/<version>/psc/<existing-entry-name>`
"Get consumer documentation for a published software service template" on page 149	`GET /zosmf/provisioning/rest/<version>/psc/<existing-entry-name>/documentation/consumer`
"Get prompt variables for a published software service template" on page 151	`GET /zosmf/provisioning/rest/<version>/psc/<existing-entry-name>/prompt-variables`
"List the published software service templates" on page 154	`GET /zosmf/provisioning/rest/<version>/psc/`

Authorization requirements

Use of the published software service template services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

The specific requirements for each published software service template service are described in the topic for that service. For an overview of the security requirements for cloud provisioning roles, see "Authorization requirements" on page 47. For details, see *IBM z/OS Management Facility Configuration Guide*.

Error response content

For the 4*nn* HTTP error status codes, additional diagnostic information beyond the HTTP status code is provided in the response body for the request. This information is provided in the form of a JSON object containing the following fields:

Table 90. Response from a software services template request failure

Field	Type	Description
`http-status`	String	HTTP status code.

Table 90. Response from a software services template request failure (continued)

Field	Type	Description
`request-method`	String	HTTP request method.
`request-uri`	String	HTTP request URI.
`reason`	String	HTTP status reason code.
`message`	String	Message describing the error.
`detailed-message`	String	Message describing the error in more detail.
`debug`	String	Debug information about for the error.

Error logging

Errors from the software services template services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 201 Created
The request succeeded and resulted in the creation of an object.

HTTP 202 Accepted
The request was successfully validated and is performed asynchronously.

HTTP 204 No content
The request succeeded, but no content is available to be returned.

HTTP 400 Bad request
The request contained incorrect parameters.

HTTP 401 Unauthorized
The request cannot be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both. Or, the client did not authenticate to z/OSMF by using a valid WWW-Authenticate header.

HTTP 404 Not found
The requested resource does not exist.

HTTP 409 Request conflict
The request cannot be processed because of conflict in the request, such as an edit conflict between multiple updates.

Related information

The run operation for a published template creates a workflow, starts the workflow, and creates a corresponding software services instance in the software services registry. To work with a software services instance, use the REST APIs described in “Software services instance services” on page 157.

Run a published software service template

Use this operation to run a software services template that is in the published state.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/psc/<name>/actions/run
```

In this request:

<version>
Is the URI path variable *<version>* that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<name>
Identifies the software services template to be run.

Query parameters

None.

Description

This operation creates a workflow, starts the workflow, and creates a corresponding software services instance in the software services registry, with a state of being-provisioned.

To work with a software services instance, use the REST APIs described in "Software services instance services" on page 157.

Request content

The request content is expected to contain a JSON object as described in Table 91 and Table 92 on page 141.

Table 91. Request content for the run software services template request

Field name	Type	Required or optional	Description
input-properties	Array of Objects	Optional	An array of required runtime property objects
domain-name	String	Optional	Required if the user has consumer authorization to more than one domain with this template name.
tenant-name	String	Optional	Required if the user has consumer authorization to more than one tenant in the same domain that contains this template name.
user-data-id	String	Optional	ID for the user data specified with user-data. Passed into the software services registry.
account-info	String	Optional	Account information to use in the JCL JOB statement. The default is the account information that is associated with the resource pool for the tenant.
user-data	String	Optional	User data that is passed into the software services registry. Can be specified only if user-data-id is provided.

Table 92. Run-Time Property

Field	Type	Description
name	String	Name of the required runtime property.
value	String	Value of the required runtime property.

Authorization requirements

The user ID must be in a tenant that the template is associated with, or be an approver.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (Normal) is returned and the response body is provided, as described in "Response content."

Response content

On successful completion, the service returns a response body, which contains a JSON object, as described in Table 93.

Table 93. Response from a run software services template version request

Field	Type	Description
`registry-info`	Object	Object mapping that matches the response body returned from a registry create action.
`workflow-info`	Object	Object mapping that matches the response body returned from a workflow create action.

Example HTTP interaction

In Figure 63, a request is submitted to run the software services template named bringUpDB2.

```
POST https://localhost:4444/zosmf/provisioning/rest/1.0/psc/bringUpDB2/actions/run


{
     "input-properties":[{"name":"CSQ_MQ_SSID","value":"ZCT1"},
     {"name":"CSQ_CMD_PFX","value":"!ZCT1"},{"name":"CSQ_ENVIRONMENT","value":"TEST"}],
     "domain-name":"default",
     "tenant-name":"tenant1"
 }
```

Figure 63. Sample request to run a software services template

The following is the response body for the request.

```
{
   "registry-info": {
     "object-name": "QMgr_7",
     "object-id": "c5a8ecdd-db35-466b-aad9-cba0f33bb84b",
     "object-uri": "/zosmf/provisioning/rest/1.0/scr/c5a8ecdd-db35-466b-aad9-cba0f33bb84b"
   },
   "workflow-info": {
     "workflowKey": "ff96459f-27fa-490a-a3e4-4086649c12f3",
     "workflowDescription": "Procedure to provision a MQ for zOS Queue Manager",
     "workflowID": "ProvisionQueueManager",
     "workflowVersion": "1.0.1",
     "vendor": "IBM"
   }
 }
```

Figure 64. Sample response body

Get a published software service template

Use this operation to retrieve a published software service template from the catalog.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/psc/<existing-entry-name>
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<existing-entry-name>
Identifies the software services template to be retrieved.

Query parameters

You can specify the following query parameter on this request. Objects matching all query parameters are returned.

domain-name
Optional, string, specifies the domain name.

If you specify no query parameters, then all templates are returned.

Description

This operation retrieves a published software service template from the catalog.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in a software services template being retrieved. A response body is provided, as described in "Response content."

Request content

None.

Authorization requirements

The user ID must be in a tenant that the template is associated with, or be an approver.

See "Authorization requirements" on page 47.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services template. See Table 95 on page 145, Table 96 on page 146, Table 97 on page 147, and Table 98 on page 147.

Table 94. Response from a get software services template request

Field	Type	Description
base-object-id	String	The object ID that is associated with all of the versions of the software services template.
generated-name	String	Generated name for the software services template.
name	String	The name associated with the software services template.
version	String	Version of the software services template.
owner	String	User ID of the software services template owner.
state	String	Indicates the status of the software services template. It is always published. The entry is locked and visible to consumers.
description	String	Description of the software services template.
tenants	Array of Strings	Each string represents a tenant that the template is associated with.
domain-name	String	The domain the template is associated with.
approvals	Array of objects	Array of Approval-Object containing information about the approvals associated with this software services template. See Table 96 on page 146.
action-definition-file	String	Location of the action definition file.
action-definition-file-original-source	String	Original user specified location of the action definition file
action-definition-file-original-timestamp	String	Last-modified time stamp for when the original action definition file source was specified, in ISO 8601 format.
actions	Array of objects	Array of Action-Object containing information about the actions associated with the template. See Table 97 on page 147.
software-id	String	A short, arbitrary, value that identifies the software that is being provisioned.
software-name	String	Name of the software that is being provisioned.
software-type	String	Type of software that is being provisioned.
software-version	String	Version of the software that is being provisioned.
workflow-definition-file	String	Location of the workflow definition file, the primary XML file that defines the workflow
workflow-definition-file-original-source	String	Original user-specified location of the workflow definition file.
workflow-definition-file-original-timestamp	String	The last-modified time stamp for when the original workflow definition file source was specified, in ISO 8601 format.
workflow-id	String	A short, arbitrary value that identifies the workflow.
workflow-vendor	String	Name of the vendor that provided the workflow definition file.
workflow-version	String	Version of the workflow definition file.
workflow-variable-input-file	String	Location of the workflow variable input file, an optional properties file used to specify in advance the values for one or more of the variables that are defined in the workflow definition file.
workflow-variable-input-file-original-source	String	The original user-specified location of the workflow variable input file.
workflow-variable-input-file-original-timestamp	String	The last-modified time stamp for when the original variable input file source was specified, in ISO 8601 format.

Table 94. Response from a get software services template request (continued)

Field	Type	Description
workflow-clean-after-provisioned	Boolean	Indicates if a workflow that performs provisioning should be automatically deleted after it completes successfully. The value is true to delete, false to keep. The default value, if none is specified, is false, which keeps the workflow.
prompt-variables	Array of objects	Array of prompt variable objects containing information about the variables that are expected to be prompted for in preparation for running the software services template. See Table 95.
at-create-variables	Array of strings	Array of strings that name the variables that are either prompt variables (variables that are expected to be prompted for in preparation for running the software services template), or required variables (variables for which a value is required when the software services template is run), or both.
consumer-documentation-file	String	Location of the original file that provides information for consumers about the template.
consumer-documentation-type	String	Type of the consumer documentation file, either text or pdf. This is required if consumer-documentation-file is specified.
admin-documentation-file	String	Location of a file that provides information for administrators about the template.
admin-documentation-type	String	Type of the administrator documentation file, either text or pdf. This is required if admin-documentation-file is specified.
create-time	String	Time that this object was created, in ISO 8601 format.
create-by-user	String	User who created this object.
last-modified-time	String	The last time this object was updated, in ISO 8601 format.
last-modified-by-user	String	User who last updated this object.

Table 95. Response from a get request: Prompt-Variable-Object

Field	Type	Description
name	String	Name of the property.
value	String	Current value for the property.
required	boolean	Indicates whether the variable value is required during the workflow create process.
label	String	Short label for the UI widget.
description	String	Explanation of what the variable is used for and perhaps what the syntactic requirements are.
abstract	String	Brief description of the variable for the UI widget.
type	String	Type of the variable element: boolean, string, integer, decimal, time, date.
must-be-choice	boolean	Indicates whether the value must come from the provided choices.
choices	Array of Strings	Contains allowable choices for the value of the variable.
regex	String	Standard regular expression that constrains the variable value.
multi-line	boolean	Indicates whether the value requires a multi-line text box.

Table 95. Response from a get request: Prompt-Variable-Object (continued)

Field	Type	Description
min	String	For a string type, indicates the minimum string length of the value. For all other types, indicates the minimum value required.
max	String	For a string type, indicates the maximum string length of the value. For all other types, indicates the maximum value required.
places	String	Maximum number of decimal places that can be specified for a variable of type decimal.
error-message	String	Default error message associated with an incorrect value.

Fields of type String default to null.

Table 96. Response from a get request: Approval-Object

Field	Type	Description
status	String	Status of the approval for this object: pending, approved, or rejected.
comment	String	Comment associated with the change in status from pending to either approved or rejected.
description	String	Additional detail provided if the approval is for a workflow definition that's associated with the action definition. For example: This workflow definition is associated with the <action-name> action.
user-ids	Array of strings	Each string in the array is a user ID. Any one of the user IDs in the array can approve or reject the item. The action of the last ID takes precedence.
status-update-by	String	The user ID that performed the last approve or reject action for this approval object
time-of-update	String	The last time this object was updated, in ISO 8601 format.
run-as-user	String	The runAsUser user ID that the approval object is for. This applies only when the type is action_definition or step_definition.
type	String	Type of approval: general (for the template), domain, action_definition, step_definition
object-id	String	Unique object id representing this approval object.
workflow-file	String	Workflow file definition associated with this runAsUser user ID. Null if the user ID is not associated with a workflow definition step or is a general approval.
variable-input-file	String	Variable input file associated with this runAsUser user ID. Null if the user ID is not associated with a workflow definition step or is a general approval.
step-name	String	Workflow file definition step associated with this runAsUser user ID. Null if the user ID is not associated with a workflow definition step or is a general approval.

Table 96. Response from a get request: Approval-Object (continued)

Field	Type	Description
called-by-step-name	String	Step in the parent workflow definition that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
called-by-workflow-file	String	Workflow definition that called the workflow definition file that generated the approval object. Used if the definition file that generated the approval object is a callable workflow.
actions-file	String	Actions file definition associated with this runAsUser user ID. Null if the user ID is not associated with an action or is a general approval.
action-name	String	Action defined in the actions file associated with this runAsUser user ID. Null if the user ID is not associated with an action or is a general approval.

Table 97. Response from a get request: Action-Object

Field	Type
name	String
type	String
command	String
workflow-definition-file	String
workflow-variable-input-file	String
workflow-variables	Variable[]
instructions	String
prompt-variables	The prompt variable objects that are associated with the action.

Table 98. Response from a get request: Variable-Object

Field	Type
name	String
value	String
visibility: public or private	String

Example HTTP interaction

In Figure 65, a request is submitted to retrieve a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/psc/bringUpDB2
```

Figure 65. Sample request to get a software services template


```
{
  "name": "mqCBA",
  "version": "1",
  "owner": "domadmin",
  "state": "published",
  "description": "This workflow provisions an MQ for z/OS Queue Manager",
  "tenants": [...],
```

```
|    "actions": [...],
|    "approvals": [],
|    "tested": false,
|    "generated-name": "mqCBA.1.default",
|    "domain-name": "default",
|    "action-definition-file": "definition/qmgrActions.xml",
|    "action-definition-file-original-source": "/users/gg/mqCBA/definition/qmgrActions.xml",
|    "action-definition-file-original-timestamp": "2016-11-18T20:00:42Z",
|    "software-id": "5655-W97",
|    "software-name": "IBM MQ for z/OS",
|    "software-type": "QMgr",
|    "software-version": "V8.0.0",
|    "workflow-definition-file": "definition/provision.xml",
|    "workflow-definition-file-original-source": "/users/gg/mqCBA/definition/provision.xml",
|    "workflow-definition-file-original-timestamp": "2016-11-18T20:03:47Z",
|    "workflow-id": "ProvisionQueueManager",
|    "workflow-vendor": "IBM",
|    "workflow-version": "1.0.1",
|    "workflow-variable-input-file": "definition/workflow_variables.properties",
|    "workflow-variable-input-file-original-source": "/users/gg/mqCBA/definition/workflow_variables.properties",
|    "workflow-variable-input-file-original-timestamp": "2016-11-18T20:00:42Z",
|    "prompt-variables": [],
|    "at-create-variables": [],
|    "workflow-clean-after-provisioned": true,
|    "security-wf-info": null,
|    "create-time": "2016-11-18T20:00:43.504Z",
|    "created-by-user": "domadmin",
|    "last-modified-by-user": "domadmin",
|    "last-modified-time": "2016-11-18T20:04:50.913Z",
|    "admin-documentation-file-original-source": "/users/gg/mqCBA/documentation/admin-mqaas_readme.pdf",
|    "admin-documentation":
|       "/zosmf/provisioning/rest/1.0/scc/5b0c3367-b856-4727-99ac-f9a79c9abf28/documentation/admin",
|    "admin-documentation-type": "pdf",
|    "consumer-documentation-file-original-source":
|       "/users/gg/mqCBA/documentation/consumer-workflow_variables.properties",
|    "consumer-documentation":
|       "/zosmf/provisioning/rest/1.0/scc/5b0c3367-b856-4727-99ac-f9a79c9abf28/documentation/consumer",
|    "consumer-documentation-type": "text",
|    "base-object-id": "c0e4d08f-f046-4a79-8a15-6981743d07ed",
|    "admin-documentation-mime-type": "application/pdf",
|    "consumer-documentation-mime-type": "text/plain"
| }
```


| e following is the response body for the example GET request.

Get consumer documentation for a published software service template

Use this operation to retrieve the consumer documentation for a published software service template from the catalog.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/psc/<existing-entry-name>/documentation/
consumer
```

In this request:

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

<existing-entry-name>
Identifies the software services template to be retrieved.

documentation/consumer
Causes the consumer documentation file to be retrieved.

Query parameters

None.

Description

This operation retrieves the consumer documentation for a published software service template from the catalog.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in the consumer documentation for a software services template being retrieved.

Request content

None.

Authorization requirements

The user ID must be in a tenant that the template is associated with, or be an approver.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

None.

Example HTTP interaction

In Figure 66 on page 150, a request is submitted to retrieve consumer documentation for a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/psc/bringUpDB2/documentation/consumer
```

Figure 66. Sample request to get consumer documentation for a software services template

Get prompt variables for a published software service template

Use this operation to retrieve the name/value pairs for variables that are required to run the software services template and for which a prompt can be used to obtain the value.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/psc/<existing-entry-name>/
prompt-variables
```

In this request:

<existing-entry-name>
Identifies the software services template for which variables are to be retrieved.

<version>
Is the URI path variable that identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

You can specify the following query parameter on this request to limit the software services templates that are returned. To be returned, a software services template must all query parameters.

`domain-name`
Optional, specifies the domain name.

Description

This operation retrieves the variables for which a prompt can obtain the value.

On successful completion, HTTP status code `200` (Normal) is returned, indicating that the request resulted in a software services template being retrieved. A response body is provided, as described in "Response content"

Request content

None.

Authorization requirements

The user ID must be in a tenant that the template is associated with, or be an approver.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the prompt variables. Table 99 lists the fields in the JSON object.

Table 99. Response from a get prompt variables request

Field	Type	Description
prompt-variables	Array of objects	An array of required run-time property objects. See Table 100 on page 152.

Table 100. Response from a get request: Prompt-Variable-Object

Field	Type	Description
name	String	Name of the property.
value	String	Current value for the property.
required	boolean	Indicates whether the variable value is required during the workflow create process.
label	String	Short label for the UI widget.
description	String	Explanation of what the variable is used for and perhaps what the syntactic requirements are.
abstract	String	Brief description of the variable for the UI widget.
type	String	Type of the variable element: boolean, string, integer, decimal, time, date.
must-be-choice	boolean	Indicates whether the value must come from the provided choices.
choices	Array of Strings	Contains allowable choices for the value of the variable.
regex	String	Standard regular expression that constrains the variable value.
multi-line	boolean	Indicates whether the value requires a multi-line text box.
min	String	For a string type, indicates the minimum string length of the value. For all other types, indicates the minimum value required.
max	String	For a string type, indicates the maximum string length of the value. For all other types, indicates the maximum value required.
places	String	Maximum number of decimal places that can be specified for a variable of type decimal.
error-message	String	Default error message associated with an incorrect value.

Example HTTP interaction

Figure 67 shows a request to retrieve the prompt variables for a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/psc/mq/prompt-variables
```

Figure 67. Sample request to get the prompt variables for a published software service template

The following is the response body for the request.

```
{
   "prompt-variables": [
     {
       "name": "CSQ_MQ_SSID",
       "label": "MQ_SSID",
       "description": "The name of the MQ subsystem to be provisioned.",
       "type": "string",
       "value": "ZCT1",
       "required": true,
       "choices": null,
       "regex": "[A-Z0-9]{1,4}",
       "min": null,
       "max": null,
       "places": null,
       "abstract": "Subsystem identifier",
       "multi-line": false,
       "must-be-choice": false,
       "error-message": "The value entered is not valid."
     },
     {
       "name": "CSQ_CMD_PFX",
       "label": "CMD_PFX",
       "description": "The MQ subsystem command prefix.",
       "type": "string",
       "value": "!ZCT1",
       "required": true,
       "choices": null,
       "regex": "['\\sa-zA-Z0-9.,!?()*+-=|;%_?:$@#&<>]{1,8}",
       "min": null,
       "max": null,
       "places": null,
       "abstract": "Command prefix",
       "multi-line": false,
       "must-be-choice": false,
       "error-message": "The value entered is not valid."
     },
     {
       "name": "CSQ_ENVIRONMENT",
       "label": "ENVIRONMENT",
       "description": "The environment for which the queue manager is to be provisioned/de-provisioned.
        The BSDS, Log and Pageset datasets vary depending on the environment.",
       "type": "string",
       "value": "TEST",
       "required": true,
       "choices": [
         "DEV",
         "TEST",
         "QA",
         "PROD"
       ],
       "regex": null,
       "min": null,
       "max": null,
       "places": null,
       "abstract": "Environment for which the queue manager is to be provisioned/de-provisioned
        (DEV, TEST, QA, PROD)",
       "multi-line": false,
       "must-be-choice": true,
       "error-message": "The value entered is not valid."
     }
   ]
 }
```

List the published software service templates

Use this operation to list the software services templates in the catalog that are in the published state.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/psc/
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software services template service. The following value is valid: `1.0`.

Query parameters

You can specify the following query parameter on this request to limit the software services templates that are returned. To be returned, a software services template must all query parameters.

`name`
Optional, regular expression, specifies the external name of the software services template.

`owner`
Optional, specifies the user ID or group ID that identifies the owner of the software services template.

`software-type`
Optional, specifies the type of software being provisioned.

`domain-name`
Optional, specifies the domain name.

Description

This operation retrieves software services templates that are in the published state from the catalog.

On successful completion, HTTP status code `200` (OK) is returned, indicating that the request resulted in software services templates being retrieved. A response body is provided, as described in "Response content."

Request content

None.

Authorization requirements

The user ID must be in a tenant that the template is associated with, or be an approver.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the software services templates. See Table 101 on page 155 and Table 102 on page 155.

Table 101. Array of objects

Field	Type	Description
psc-list	Array of objects	Array of software services template objects. The array is filtered based on any query parameters that were provided.

Table 102. Fields for each software services template

Field	Type	Description
generated-name	String	The generated name for the software services template.
object-id	String	The ID that identifies the software services template.
base-object-id	String	The object ID that is associated with all the versions of the template. software services template.
name	String	Descriptive name for the software services template. The name must be unique. It can be up to 100 characters. The name cannot contain the symbols for less than (<), greater than (>), or ampersand (&).
version	String	Version of the software services template.
owner	String	User ID of the software services template owner.
state	String	Indicates the current status of the software services template: **published** The entry is locked and visible in the marketplace.
description	String	Description of the software services template.
domain-name	String	Name of the domain this template resides in.
action-definition-file	String	Location of the action definition file.
software-id	String	A short, arbitrary, value that identifies the software being provisioned.
software-name	String	Name of the software being provisioned.
software-type	String	Identifies the type of software being provisioned.
software-version	String	Version of the software being provisioned.
workflow-definition-file	String	Location of the workflow definition file for the software services template. This file is the primary XML file for the workflow definition.
workflow-id	String	Identifies the workflow.
workflow-vendor	String	Name of the vendor that provided the workflow definition file.
workflow-version	String	Version of the workflow definition file.
workflow-variable-input-file	String	Optional properties file that specifies values for one or more of the variables that are defined in the workflow definition file.
create-time	String	The time that this software services template was created, in ISO 8601 format.
create-by-user	String	The user that created this software services template.
last-modified-time	String	The last time this software services template was updated, in ISO 8601 format.
last-modified-by-user	String	The user that last updated this software services template.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 103. Response from a software services template request failure

Field	Type	Description
http-status	String	HTTP status code.
request-method	String	HTTP request method.
request-uri	String	HTTP request URI.
reason	String	HTTP status reason code.
message	String	Message describing the error.
detailed-message	String	Message describing the error in more detail.
debug	String	Debug information about for the error.

Example HTTP interaction

Figure 68 shows a request to retrieve a software services template.

```
GET https://localhost:4444/zosmf/provisioning/rest/1.0/psc HTTP/1.1
```

Figure 68. Sample request to list all published software service templates

The following is the response body for the request.

```
{
  "psc-list": [
    {
      "name": "mqCBA",
      "version": "1",
      "owner": "domadmin",
      "state": "published",
      "description": "This workflow provisions an MQ for z/OS Queue Manager",
      "generated-name": "mqCBA.1.default",
      "object-id": "5b0c3367-b856-4727-99ac-f9a79c9abf28",
      "base-object-id": "c0e4d08f-f046-4a79-8a15-6981743d07ed",
      "domain-name": "default",
      "action-definition-file": "definition/qmgrActions.xml",
      "software-id": "5655-W97",
      "software-name": "IBM MQ for z/OS",
      "software-type": "QMgr",
      "software-version": "V8.0.0",
      "workflow-definition-file": "definition/provision.xml",
      "workflow-id": "ProvisionQueueManager",
      "workflow-vendor": "IBM",
      "workflow-version": "1.0.1",
      "workflow-variable-input-file": "definition/workflow_variables.properties",
      "create-time": "2016-11-18T20:00:43.504Z",
      "created-by-user": "domadmin",
      "last-modified-by-user": "domadmin",
      "last-modified-time": "2016-11-18T20:28:43.951Z"
    }
  ]
}
```

Figure 69. Response body for the GET request

Software services instance services

The software services instance services are application programming interfaces (APIs), which are implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to create and manage software services instances in the software services registry.

For information about cloud provisioning, including a description of the roles, see "Cloud provisioning services" on page 46.

The software services registry contains a list of the software on z/OS that has been registered as being provisioned, typically through the use of software services templates, although provisioning can be done manually. It is maintained in the z/OSMF data repository and has a sysplex-wide scope.

Table 104 lists the operations that the software services instance services provide.

Table 104. z/OSMF software services instance services: operations summary

Operation name	HTTP method and URI path
"Create a software services instance" on page 160	`POST /zosmf/provisioning/rest/<version>/scr`
"Get the contents of a software services instance" on page 167	`GET /zosmf/provisioning/rest/<version>/scr/<object-id>`
"Get the variables for a software services instance" on page 179	`GET /zosmf/provisioning/rest/<version>/scr/<object-id>/variables`
"Get key-value variables for a software services instance" on page 182	`GET /zosmf/provisioning/rest/<version>/scr/<object-id>/key-value-variables`
"List the software services instances" on page 174	`GET /zosmf/provisioning/rest/<version>/scr`
"Perform an action against a software services instance" on page 193	`POST /zosmf/provisioning/rest/<version>/scr/<object-id>/actions/<action>`
"Delete a software services instance" on page 191	`DELETE /zosmf/provisioning/rest/<version>/scr/<object-id>`
"Update a software services instance" on page 184	`PUT /zosmf/provisioning/rest/<version>/scr/<object-id>`
"Update variables in a software services instance" on page 189	`PUT /zosmf/provisioning/rest/version/scr/object-id/variables`
"Get the response for an action performed against a software services instance" on page 196	`GET /zosmf/provisioning/rest/<version>/scr/<object-id>/actions/<action-id>`
"List the responses for actions performed against a software services instance" on page 199	`GET /zosmf/provisioning/rest/<version>/scr/<object-id>/actions`

Table 104. z/OSMF software services instance services: operations summary (continued)

Operation name	HTTP method and URI path
"Delete the response for an action performed against a software services instance" on page 202	`DELETE /zosmf/provisioning/rest/<version>/scr/<object-id>/actions/<action-id>`

Authorization requirements

Use of the software services instance services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

In addition, the user's z/OS user ID may need access to other resources, including those that define roles such as the domain administrator. The specific requirements for each software services instance service are described in the topic for that service. For an overview of the security requirements for cloud provisioning roles, see "Authorization requirements" on page 47. For details, see *IBM z/OS Management Facility Configuration Guide*.

Error response content

For the 4*nn* HTTP error status codes, additional diagnostic information beyond the HTTP status code is provided in the response body for the request. This information is provided in the form of a JSON object containing the following fields:

Table 105. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Error logging

Errors from the software services instance services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 201 Created
The request succeeded and resulted in the creation of an object.

HTTP 202 Accepted
The request was successfully validated and is performed asynchronously.

HTTP 204 No content
The request succeeded, but no content is available to be returned.

HTTP 400 Bad request
The request contained incorrect parameters.

HTTP 401 Unauthorized
The request cannot be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both. Or, the client did not authenticate to z/OSMF by using a valid WWW-Authenticate header.

HTTP 404 Not found
The requested resource does not exist.

HTTP 409 Request conflict
The request cannot be processed because of conflict in the request, such as an edit conflict between multiple updates.

Create a software services instance

You can use this operation to create a software services instance in the registry.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scr
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software services instance service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates a software services instance in the registry, based on the properties that are specified in the request body (a JSON object). For the properties that you can specify, see "Request content."

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the creation of a new software services instance. The URI path for the software object is provided in the Location response header and a response body is provided, as described in "Response content" on page 164.

Request content

The request content is expected to contain a JSON object. Table 106 lists the fields in the JSON object.

Table 106. Request content for the create software services instance request

Field name	Type	Required or optional	Description
type	String	Required	Type of the software. Up to 8 characters.
registry-type	String	Required	The type of software registry object: catalog or general. An object with registry-type catalog is created from a software services template in the software services catalog. When the registry type is catalog, a catalog object ID and catalog name are also required.
state	String	Required	The current state of the software: • being-initialized • being-provisioned • provisioned • being-deprovisioned • deprovisioned • provisioning-failed • deprovisioning-failed
catalog-object-id	String	Required when registry-type is catalog	The identifier of the software services template to be used to create the software services instance.

Table 106. Request content for the create software services instance request (continued)

Field name	Type	Required or optional	Description
catalog-object-name	String	Required when registry-type is catalog	The name of the software services template to be used to create the software services instance.
external-name	String	Optional	The external name to identify the software registry object. If the external name is not provided then it is set from object-name in the response body. Up to 34 characters.
system-nickname	String	Required when registry-type is catalog	The name of the system entry in the system entry table of the software.
system	String	Optional	System on which the software is provisioned. Up to 8 characters.
sysplex	String	Optional	Sysplex on which the software is provisioned. Up to 8 characters.
vendor	String	Optional	Vendor of the software. Up to 24 characters.
version	String	Optional	Version of the software. Up to 24 characters.
description	String	Optional	Description for the software. Up to 256 characters.
owner	String	Optional	The user ID that identifies the owner of the software registry object. Up to 8 characters.
provider	String	Optional	The user ID that identifies the provider of the software, . Up to 8 characters. This is the owner of the software catalog object.
quality-attributes	String	Optional	The quality attributes associated with the software. Up to 16 characters.
workflow-key	String	Optional	The workflow key associated with provisioning the software. This field is not valid when the value for registry-type is general.
workflow-clean-after-provisioned	String	Optional	The indication of whether the workflow instance used to provision this instance will be removed after the workflow is completed. Must be true or false. This field is not valid when the value for registry-type is general.
actions	Action[]	Optional	The actions for the software. See Table 107 on page 162.
variables	Variable[]	Optional	The variables for the software. See Table 108 on page 163.
user-data-id	String	Optional	The user data ID.
user-data	String	Optional	The user data.
domain-id	String	See description.	The domain ID. This field is not valid when the value for registry-type is general. It is required when the value for registry-type is catalog.
tenant-id	String	See description.	The tenant ID. This field is not valid when the value for registry-type is general. It is required when the value for registry-type is catalog.
domain-name	String	See description.	The name of the domain. This field is not valid when the value for registry-type is general. It is required when the value for registry-type is catalog.
tenant-name	String	See description.	The name of the tenant. This field is not valid when the value for registry-type is general. It is required when the value for registry-type is catalog.

Table 106. Request content for the create software services instance request (continued)

Field name	Type	Required or optional	Description
ssin	String	Optional	Software service instance name, used in generating names for software services instances. This field is not valid when the value for registry-type is general.

Table 107. Action structure

Field	Type	Description
name	String	The name of the action. If the name of the action is deprovision, the action is for deprovisioning the software. You can indicate that the action is for deprovisioning either by setting the is-deprovision field to `true` or by naming the action `deprovision`.
type	String	The type of the action. The value must be one of the following: • command • workflow • instructions
is-deprovision	String	Indicates if the action deprovisions the software, as follows: • If true, the action deprovisions the software. • If false or not set, the action does not deprovision the software. This is overriden if the value of the name field is deprovision.
command	String	For command type actions, the command.
command-run-as-user	String	For command type actions, if provided, the user ID to be used when the command is run. This is not valid when the registry-type is general.
command-sol-key	String	For command type actions, if provided, the key to search for in the solicited messages command response.
command-unsol-key	String	For command type actions, if provided, the key to search for in the unsolicited messages.
command-detect-time	String	For command type actions, if provided, the time in seconds to detect for the command-unsol-key in the unsolicited messages. Also, the minimum time before a command response is checked for after the command is submitted for execution. If not provided, the default command-detect-time is 15 seconds when the command-unsol-key is specified or 10 seconds when the command-unsol-key is not specified.
workflow-definition-file	String	For workflow type actions, the workflow definition file.
workflow-variable-input-file	String	For workflow type actions, if provided, the workflow variable input file.
workflow-variables	Variable[]	For workflow type actions, if provided, the workflow variables.
instructions	String	For instruction type actions, the instructions.
prompt-variables	PromptVariable[]	For workflow type actions, if provided, the prompt variables, which are the variables that will have their values prompted for at create time. See Table 109 on page 163.

Table 107. Action structure (continued)

Field	Type	Description
at-create-variables	String[]	For workflow type actions, if provided, the names of the at create variables, which are the only variables allowed on input-variables for the do action operation.

Table 108. Variable structure

Field	Type
name	String
value	String
visibility	String. The value must be public or private.

Table 109. Prompt variable structure

Field	Type
name	String
value	String
abstract	String
description	String
error-message	String
label	String
max	String
min	String
milti-line	Boolean
must-be-choice	Boolean
choices	String List
places	String
regex	String
required	Boolean
type	String

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content" on page 164.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 110. HTTP error response codes for a create software services instance request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.

Response content

On successful completion, the service returns the following:

- URI path of the created software services instance in the Location response header.
- Response body, which contains a JSON object with details about the software services instance. Table 111 lists the fields in the JSON object.

Table 111. Response from a create software services instance request

Field	Type	Description
`object-id`	String	The object ID of the newly created object. The object ID is to be used on further requests to the object.
`object-uri`	String	The object URI of the newly created object.
`object-name`	String	The object name of the newly created object.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 112. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In Figure 70 on page 165, a request is submitted to create a software services instance on the system `SY1`.

```
{
 "type":"DB2",
 "external-name":"DB2B",
 "vendor":"IBM",
 "version":"V5R10",
 "description":"DB2 for test1",
 "registry-type":"catalog",
 "catalog-object-id":"9f7c659e-38f5-4585-b9f9-9cd448bf9cc3",
 "catalog-object-name":"DB2template1",
 "workflow-key":"02e1ec78-e0db-482b-8013-3d435b52e2e3",
 "workflow-clean-after-provision":"true",
 "system-nickname":"SYSTEM1",
 "system":"SY1",
 "sysplex":"PLEX1",
 "state":"being-provisioned",
 "owner":"ZOSMFAD",
 "provider":"ZOSMFAD",
 "quality-attributes":"123456789ABCDEF0",
 "user-data":"my data",
 "user-data-id":"udid1",
 "domain-id":"izu$0",
 "tenant-id":"izu$002",
 "domain-name":"default",
 "tenant-name":"default",
 "ssin":"SSIN1",
 "variables":
     [
     {
         "name":"IACTION_NAME",
         "value":"Instructions1",
         "visibility":"public"
     },
     {
         "name":"COMMAND1",
         "value":"d a,l",
         "visibility":"public"
     },
     {
         "name":"C_DETECT_TIME",
         "value":"25",
         "visibility":"public"
     },
     {
         "name":"C_SOL_K",
         "value":"VLF",
         "visibility":"public"
     },
     {
         "name":"C_UNSOL_K",
         "value":"CSV",
         "visibility":"public"
     }
     ],
 "actions":
     [
     {
         "name":"Instructions1",
         "type":"instructions",
         "is-deprovision":"false",
         "instructions":"These are the instructions for the ${IACTION_NAME} action."
     },
     {
         "name":"command1",
         "type":"command",
         "is-deprovision":"false",
         "command":"${COMMAND1}",
         "command-detect-time":"${C_DETECT_TIME}",
         "command-run-as-user":"IBMUSER",
         "command-sol-key":"${C_SOL_K}"
         ,"command-unsol-key":"${C_UNSOL_K}"
     },
     {
         "name":"deprovision",
         "type":"instructions",
         "is-deprovision":"true",
         "instructions":"Do the deprovision manually."
     }
     ]
 }
```

Figure 70. Sample request to create a software services instance

The response is shown below.

```
{
"object-name": "DB2_1",
"object-id": "c7156cbf-e1ce-4f05-b7c7-96d73dfb94f9",
"object-uri": "/zosmf/provisioning/rest/1.0/scr/c7156cbf-e1ce-4f05-b7c7-96d73dfb94f9"
}
```

Get the contents of a software services instance

You can use this operation to retrieve the contents of a software services instance.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scr/<object-id>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be retrieved.

Query parameters

None.

Description

This operation retrieves the properties of a software services instance.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 114 on page 168.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For catalog registry type objects, the user issuing the request must be at least one of the following:

- The owner of the software services instance
- A member of the tenant of the software services instance
- A member of a tenant that was given access to the software services instance
- A domain administrator of the software services instance.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 114 on page 168.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 113. HTTP error response codes for a get software services instance contents request

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance was not found; the software services instance does not exist.

Response content

On successful completion, the response body is a JSON object that contains the retrieved data. Table 114 lists the fields in the JSON object.

Table 114. JSON object that is returned for a get software services instance property request

Field	Type	Description
object-id	String	The object-id for the software services instance
object-name	String	The object-name for the software services instance
type	String	Type of the software
registry-type	String	Type of registry object: catalog or general
external-name	String	External name of the software services instance
system-nickname	String	The name of the system entry in the system entry table of the software.
system	String	System on which the software is provisioned
sysplex	String	Sysplex on which the software is provisioned
vendor	String	Vendor of the software
version	String	Version of the software
description	String	Description for the software
owner	String	The user ID that identifies the owner of the software
provider	String	The user ID that identifies the provider of the software
catalog-object-id	String	The identifier of the template that is used when partitioning the software represented by this instance. Only valid when registry-type is catalog.
catalog-object-name	String	The name of the template that was used when partitioning the software represented by this instance.
workflow-key	String	The workflow key that is associated with provisioning the software.
workflow-clean-after-provisioned	String	The Indication of whether the workflow instance used to provision this instance will be removed after the workflow is completed. Must be true or false.
state	String	The current state of the software: • being-initialized • being-provisioned • provisioned • being-deprovisioned • deprovisioned • provisioning-failed • deprovisioning-failed
quality-attributes	String	The quality attributes of the software
actions	Action[]	The actions for the software. Table 115 on page 169
variables	Variable[]	The variables for the software. See Table 116 on page 170.
created-time	String	The time the object was created. The time is in the ISO8601 format.
last-modified-time	String	The time the object was updated. The time is in the ISO8601 format.
created-by-user	String	The user ID that created the object

Table 114. JSON object that is returned for a get software services instance property request (continued)

Field	Type	Description
last-modified-by-user	String	The user ID that last updated the object
last-action-name	String	The name of the last action that was performed.
last-action-object-id	String	The action ID of the last action that was performed.
last-action-state	String	The state of the last action that was performed.
user-data-id	String	The user data ID.
user-data	String	The user data.
domain-id	String	The domain ID.
tenant-id	String	The tenant ID.
domain-name	String	The name of the domain.
tenant-name	String	The name of the tenant.
ssin	String	Software service instance name, used in generating names for software services instances.

Table 115. Action structure

Field	Type	Description
name	String	The name of the action. If the name of the action is deprovision, the action is for deprovisioning the software. You can indicate that the action is for deprovisioning either by setting the is-deprovision field to `true` or by naming the action `deprovision`.
type	String	The type of the action. The value must be one of the following: • command • workflow • instructions
is-deprovision	String	Indicates if the action deprovisions the software, as follows: • If true, the action deprovisions the software. • If false or not set, the action does not deprovision the software. This is overriden if the value of the name field is deprovision.
command	String	For command type actions, the command.
command-run-as-user	String	For command type actions, if provided, the user ID to be used when the command is run. This is not valid when the registry-type is general.
command-sol-key	String	For command type actions, if provided, the key to search for in the solicited messages command response.
command-unsol-key	String	For command type actions, if provided, the key to search for in the unsolicited messages.
command-detect-time	String	For command type actions, if provided, the time in seconds to detect for the command-unsol-key in the unsolicited messages. Also, the minimum time before a command response is checked for after the command is submitted for execution. If not provided, the default command-detect-time is 15 seconds when the command-unsol-key is specified or 10 seconds when the command-unsol-key is not specified.

Table 115. Action structure (continued)

Field	Type	Description
workflow-definition-file	String	For workflow type actions, the workflow definition file.
workflow-variable-input-file	String	For workflow type actions, if provided, the workflow variable input file.
workflow-variables	Variable[]	For workflow type actions, if provided, the workflow variables.
instructions	String	For instruction type actions, the instructions.
prompt-variables	PromptVariable[]	For workflow type actions, if provided, the prompt variables, which are the variables that are expected to be prompted for in preparation for running the software services template. See Table 117.
at-create-variables	String[]	For workflow type actions, if provided, these are the names of the variables that are either prompt variables (variables that are expected to be prompted for in preparation for running the software services template), or required variables (variables for which a value is required when the software services template is run), or both. These are the only variables allowed on input-variables for the do action operation.

Table 116. Variable structure

Field	Type
name	String
value	String
visibility	String. The value must be public or private.

Table 117. Prompt variable structure

Field	Type
name	String
value	String
abstract	String
description	String
error-message	String
label	String
max	String
min	String
milti-line	Boolean
must-be-choice	Boolean
choices	String List
places	String
regex	String
required	Boolean
type	String

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 118. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the GET method is used to retrieve information about a software services instance. The software services instance is uniquely identified by the software services instance key, which is represented by the following string value: `76963ea5-81a4-42d6-99d6-f3e19747cf61`.

```
GET /zosmf/provisioning/rest/ 1.0/scr/76963ea5-81a4-42d6-99d6-f3e19747cf61
```

Figure 71. Sample request to get software services instance properties

The following is an example of the response.

```
{
  "type":"DB2",
  "system-nickname":"SYSTEM1",
  "system":"SY1",
  "sysplex":"PLEX1",
  "vendor":"IBM",
  "version":"V5R10",
  "description":"DB2 for test1",
  "owner":"ZOSMFAD",
  "provider":"ZOSMFAD",
  "state":"provisioned",
  "object-id":"c7156cbf-e1ce-4f05-b7c7-96d73dfb94f9",
  "object-name":"DB2_1",
  "object-uri":"/zosmf/provisioning/rest/1.0/scr/c7156cbf-e1ce-4f05-b7c7-96d73dfb94f9",
  "registry-type":"catalog",
  "external-name":"DB2B",
  "catalog-object-id":"9f7c659e-38f5-4585-b9f9-9cd448bf9cc3",
  "catalog-object-name":"DB2template1",
  "quality-attributes":"123456789ABCDEF0",
  "user-data":"my data",
  "user-data-id":"udid1",
  "domain-id":"izu$0",
  "tenant-id":"izu$002",
  "domain-name":"default",
  "tenant-name":"default",
  "ssin":"SSIN1",
  "workflow-key":"02e1ec78-e0db-482b-8013-3d435b52e2e3",
  "workflow-clean-after-provision":"true",
  "variables":
  [
  {
    "name":"IACTION_NAME",
    "value":"Instructions1",
    "visibility":"public"
  },
  {
    "name":"COMMAND1",
    "value":"d a,l",
    "visibility":"public"
  },
  {
    "name":"C_DETECT_TIME",
    "value":"25",
    "visibility":"public"
  },
  {
    "name":"C_SOL_K",
    "value":"VLF",
    "visibility":"public"
  },
  {
    "name":"C_UNSOL_K",
    "value":"CSV",
    "visibility":"public"
  }
 ],
  "actions":
  [
  {
    "name":"Instructions1",
    "type":"instructions",
    "is-deprovision":"false",
    "command":null,
    "instructions":"These are the instructions for the ${IACTION_NAME} action.",
    "command-run-as-user":null,
    "command-sol-key":null,
    "command-unsol-key":null,
    "command-detect-time":null,
    "workflow-definition-file":null,
    "workflow-variable-input-file":null,
    "workflow-clean-after-complete":"null",
    "variables":null,
    "at-create-variables":null,
    "prompt-variables":null
  },
```


Figure 72. Sample response from a get request (part 1 of 2)

```
{
    "name":"command1",
    "type":"command",
    "is-deprovision":"false",
    "command":"${COMMAND1}",
    "instructions":null,
    "command-run-as-user":"IBMUSER",
    "command-sol-key":"${C_SOL_K}",
    "command-unsol-key":"${C_UNSOL_K}",
    "command-detect-time":"${C_DETECT_TIME}",
    "workflow-definition-file":null,
    "workflow-variable-input-file":null,
    "workflow-clean-after-complete":"null",
    "variables":null,
    "at-create-variables":null,
    "prompt-variables":null
  },
  {
    "name":"deprovision",
    "type":"instructions",
    "is-deprovision":"true",
    "command":null,
    "instructions":"Do the deprovision manually.",
    "command-run-as-user":null,
    "command-sol-key":null,
    "command-unsol-key":null,
    "command-detect-time":null,
    "workflow-definition-file":null,
    "workflow-variable-input-file":null,
    "workflow-clean-after-complete":"null",
    "variables":null,
    "at-create-variables":null,
    "prompt-variables":null
  }
  ],
  "created-time":"2015-12-24T15:18:51.903Z",
  "last-modified-time":"2015-12-24T15:23:43.071Z",
  "created-by-user":"ZOSMFAD",
  "last-modified-by-user":"ZOSMFAD",
  "last-action-name": "Instructions1",
  "last-action-object-id": "672c54bd-6c2b-49fd-ad3b-2ec027f54089",
  "last-action-state": "complete"
}
```

Figure 73. Sample response from a get request (part 2 of 2)

List the software services instances

You can use this operation to list the software services instances in the software services registry.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scr
```

In this request, the URI path variable is as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.

Query parameters

You can specify the following query parameter on this request. Objects matching all query parameters are returned.

type
Optional, specifies the type of the software.

registry-type
Optional, specifies the type of the registry object: catalog or general.

object-name
Optional, regular expression, specifies the name for the software services instance.

external-name
Optional, regular expression, specifies the external name of the software.

system
Optional, specifies the system on which the software is provisioned.

sysplex
Optional, specifies the sysplex on which the software is provisioned.

vendor
Optional, specifies the vendor of the software.

owner
Optional, specifies the user ID that identifies the owner of the software.

provider
Optional, specifies the user ID that identifies the provider of the software.

state
Optional, specifies the current state of the software:

- being-initialized
- being-provisioned
- provisioned
- being-deprovisioned
- deprovisioned
- provisioning-failed
- deprovisioning-failed

catalog-object-id
Optional, specifies the catalog object ID associated with the creation of this software services instance.

user-data-id
Optional, specifies the ID for the user data.

domain-id
Optional, specifies the ID of the domain.

tenant-id
Optional, specifies the ID of the tenant.

tenant-name
Optional, regular expression that specifies the name of the tenant.

domain-name
Optional, regular expression that specifies the name of the domain.

If you specify no query parameters, then all objects are returned.

Description

This operation returns the object ID and a subset of fields for software services instances.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in "Response content."

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For catalog registry type objects, the user that issues the request must be at least one of the following for a software services instance to be returned in the list:

- The owner of the software services instance
- A member of the tenant of the software services instance
- A domain administrator of the software services instance.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in "Response content."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 119. HTTP error response codes for a list software services instances request.

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.

Response content

On successful completion, the response body contains a JSON object, named scr-list, of registry objects that consist of a subset of the fields for all software services instances matching the query. Table 120 on page 176 lists the fields in the JSON object.

Table 120. JSON object that is returned for a list software services instances request.

Field	Type	Description
object-id	String	The object ID for the software services instance
object-name	String	The object name for the software services instance
type	String	The type of the software, for example, the subsystem name, such as DB2 or MQ
registry-type	String	Type of registry object: catalog or general
external-name	String	External name of the software services instance
catalog-object-id	String	The identifier of the software services template used when partitioning the software represented by this software services instance. Valid only when the value for registry-type is catalog.
catalog-object-name	String	The name of the template that was used when partitioning the software represented by this instance.
system	String	Name specified for the system on the SYSNAME parameter in the IEASYSxx parmlib member.
sysplex	String	Name of the sysplex where the z/OS system is a member. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.
vendor	String	Vendor of the software
version	String	Version of the software
description	String	Description for the software
owner	String	The user ID that identifies the owner of the software
provider	String	The user ID that identifies the provider of the software
state	String	The current state of the software: • being-initialized • being-provisioned • provisioned • being-deprovisioned • deprovisioned • provisioning-failed • deprovisioning-failed
created-time	String	The time the object was created. The time is in the ISO8601 format.
last-modified-time	String	The time the object was updated. The time is in the ISO8601 format.
created-by-user	String	The user ID that created the object
last-updated-by-user	String	The user ID that last updated the object
last-action-name	String	The name of the last action that was performed.
last-action-object-id	String	The action ID of the last action that was performed.
last-action-state	String	The state of the last action that was performed.
user-data-id	String	The user data ID.
tenant-id	String	The tenant ID.
domain-id	String	The domain ID.
tenant-name	String	The name of the tenant.
domain-name	String	The name of the domain.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 121. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In Figure 74, the GET method is used to list the software services instances.

```
GET /zosmf/provisioning/rest/ 1.0/scr
```

Figure 74. Sample request to list software services instances

Figure 75 on page 178 shows the response.

```
{"scr-list":
                          {
                          "object-id":"76963ea5-81a4-42d6-99d6-f3e19747cf61",
                          "object-name":"DB2_1",
                          "system":"SYS1",
                          "sysplex":"PLEX1",
                          "type":"DB2",
                          "vendor":"IBM",
                          "version":"V1R0",
                          "owner":"admin1",
                          "provider":"provd1",
                          "registry-type":"catalog",
    "catalog-object-id":"9f7c659e-38f5-4585-b9f9-9cd448bf9cc3",
    "catalog-object-name":"DB2template1",
                          "state":"provisioned",
    "user-data-id":"udid1",
                          "description":"DB2 for test1",
                          "created-time":"2015-06-08T14:32:42.551Z",
                          "last-modified-time":"2015-06-08T14:58:12.452Z",
                          "created-by-user":"admin1",
    "last-modified-by-user":"admin1",
    "last-action-name": "Instructions1",
    "last-action-object-id": "672c54bd-6c2b-49fd-ad3b-2ec027f54089",
    "last-action-state": "complete",
    "domain-id":"izu$0",
    "tenant-id":"izu$002",
    "tenant-name":"default",
    "domain-name":"default"
                          },
                          {
                          "object-id":"cb98c8b1-9753-4e5f-86cb-169565b4f606",
                          "object-name":"DB2_2",
                          "system":"SYS1",
                          "sysplex":"PLEX1",
                          "type":"DB2",
                          "vendor":"IBM",
                          "version":"V1R0",
                          "owner":"admin1",
                          "provider":"provd1",
                          "registry-type":"catalog",
    "catalog-object-id":"9f7c659e-38f5-4585-b9f9-9cd448bf9cc3",
   "catalog-object-name":"DB2template1",
                          "state":"being-provisioned",
    "user-data-id":"udid2",
    "description":"DB2 for test2",
                          "created-time":"2015-06-08T17:26:17.128Z",
                          "last-modified-time":"2015-06-08T17:26:17.128Z",
                          "created-by-user":"admin1",
    "last-modified-by-user":"admin1",
    "last-action-name": "Instructions2",
    "last-action-object-id": "472c54bd-6c2b-49fd-ad3b-2ec027f54080",
    "last-action-state": "complete",
    "domain-id":"izu$0"
    "tenant-id":"izu$002"
    "tenant-name":"default",
    "domain-name":"default"
                          }
}
```

Figure 75. Sample response from a list software services instances request

Get the variables for a software services instance

You can use this operation to retrieve the variables for a software services instance.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scr/<object-id>/variables
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be retrieved.

Query parameters

You can specify the following query parameter on this request:

`name`
Use this optional parameter to specify variable names. You can use regular expressions.

`visibility`
Use this optional parameter to specify the visibility of the variables.

Description

This operation retrieves the variables for a software services instance.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in "Response content" on page 180.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For catalog registry type objects, the user issuing the request must be at least one of the following:

- The owner of the software services instance
- A member of the tenant of the software services instance
- A domain administrator of the software services instance.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (Normal) is returned and the response body is provided, as described in "Response content" on page 180.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 122. HTTP error response codes for a get software services instance variables request.

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requestor user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance was not found because it does not exist.

Response content

On successful completion, the response body contains a JSON object consisting of an array of the variable names and values for the software services instance, described in the tables that follow.

Table 123. Get variables request: Format of the variables object.

Field name	Type	Description
`variables`	Array of variables	Variables for the software services instance. Refer to Table 124

Table 124. Variable structure

Field	Type
name	String
value	String
visibility	String. The value must be public or private.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 125. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the GET method is used to retrieve variables for a software object. The software services instance is uniquely identified by the software services instance key, which is represented by the following string value: `76963ea5-81a4-42d6-99d6-f3e19747cf61`.

```
GET /zosmf/provisioning/rest/1.0/76963ea5-81a4-42d6-99d6-f3e19747cf61/variables
```

Figure 76. Sample request to get software services instance variables

An example of the response is shown in the figures that follow.

```
{
      "variables":[
                     {"name":"var1","value":"val1","visibility":"public"},
                     {"name":"var2","value":"val2","visibility":"public"}
       ]
}
```

Figure 77. Sample response from a get software services instance variables request

Get key-value variables for a software services instance

You can use this operation to retrieve the variables for a software services instance in key-value format.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scr/<object-id>/key-value-variables
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be retrieved.

Query parameters

You can specify the following query parameter on this request:

name
Use this optional parameter to specify variable names. You can use regular expression.

Description

This operation retrieves the variables for a z/OSMF software services instance in key-value format.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in "Response content" on page 183.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For catalog registry type objects, the user issuing the request must be at least one of the following:

- The owner of the software services instance
- A member of the tenant of the software services instance
- A domain administrator of the software services instance.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (normal) is returned and the response body is provided, as described in "Response content" on page 183.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 126. HTTP error response codes for a get software services instance key-value variables request.

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance instance was not found; the software services instance does not exist.

Response content

On successful completion, the response body contains a JSON object that consists of an array of the variable names and values for the software services instance, described in the tables that follow.

Table 127. Get key-value variables request: Format of the variables object.

Field name	Type	Description

`variables`	List of variable names and values in key-value pair format.
	
	Example:
	`{"var1":"val1",`
	`"var2":"val2"}`
		Variables for the software services instance. Only variables with public visibility are returned.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 128. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the GET method is used to retrieve key-value variables for a software object. The software services instance is uniquely identified by the software services instance key, which is represented by the following string value: `76963ea5-81a4-42d6-99d6-f3e19747cf61`.

```
GET /zosmf/provisioning/rest/1.0/76963ea5-81a4-42d6-99d6-f3e19747cf61/key-value-variables
```

Figure 78. Sample request to get software services instance variables in key-value format

The following is an example of the response.

```
{
       "variables":{
                          {"var1":"val1",
                           "var2":"val2"}
       }
}
```

Figure 79. Sample response from a get key-value variables request

Update a software services instance

You can use this operation to update fields in a software services instance.

HTTP method and URI path

```
PUT /zosmf/provisioning/rest/<version>/scr/<object-id>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be updated.

Query parameters

None.

Description

This operation updates a software services instance.

Request content

The request content is expected to contain a JSON object containing the fields to be updated. Table 129 lists the fields that are valid.

Table 129. Request content for the update software services instance request

Field name	Type	Required or optional	Description
state	String	Optional	The current state of the software: • being-initialized • being-provisioned • provisioned • being-deprovisioned • deprovisioned • provisioning-failed • deprovisioning-failed
external-name	String	Optional	The external name to identify the software registry object. Up to 25 characters.
system	String	Optional	System on which the software is provisioned, up to eight characters. Cannot be updated if the registry type is catalog for the software services instance.
sysplex	String	Optional	Sysplex on which the software is provisioned, up to eight characters Cannot be updated if the registry type is catalog for the software services instance.
vendor	String	Optional	Vendor of the software, up to 24 characters Cannot be updated if the registry type is catalog for the software services instance.

Table 129. Request content for the update software services instance request (continued)

Field name	Type	Required or optional	Description
version	String	Optional	Version of the software, up to 24 characters Cannot be updated if the registry type is catalog for the software services instance.
description	String	Optional	Description for the software, up to 256 characters
owner	String	Optional	The user ID that identifies the owner of the software registry object, up to eight characters Cannot be updated if the registry type is catalog for the software services instance.
provider	String	Optional	The user ID that identifies the provider of the software, up to eight characters. This is the owner of the software catalog object. Cannot be updated if the registry type is catalog for the software services instance.
quality-attributes	String	Optional	The quality attributes associated with the software, up to 16 characters Cannot be updated if the registry type is catalog for the software services instance.
workflow-key	String	Optional	The workflow key associated with provisioning the software. This field is not valid when the value for registry-type is general.
workflow-clean-after-provisioned	String	Optional	The indication of whether the workflow instance used to provision this instance will be removed after the workflow is completed. Must be true or false. This field is not valid when the value for registry-type is general.
actions	Action[]	Optional	The actions for the software. Cannot be updated if the registry type is catalog for the software services instance. See Table 130 on page 186.
variables	Variable[]	Optional	The variables for the software. Refer to Table 131 on page 187. Cannot be updated if the registry type is catalog for the software services instance.
user-data-id	String	Optional	The user data ID.
user-data	String	Optional	The user data.
ssin	String	Optional	Software service instance name, used in generating names for software services instances. This field is not valid when the value for registry-type is general.

Table 130. Action structure

Field	Type	Description
name	String	The name of the action. If the name of the action is deprovision, the action is for deprovisioning the software. You can indicate that the action is for deprovisioning either by setting the is-deprovision field to `true` or by naming the action `deprovision`.
type	String must be one of: • command • workflow • instructions	The type of the action.
is-deprovision	String	Indicates if the action deprovisions the software, as follows: • If true, the action deprovisions the software. • If false or not set, the action does not deprovision the software. This is overriden if the value of the name field is deprovision.
command	String	For command type actions, the command.
command-run-as-user	String	For command type actions, if provided, the user ID to be used when the command is run.This is not valid when the registry-type is general.
command-sol-key	String	For command type actions, if provided, the key to search for in the solicited messages command response.
command-unsol-key	String	For command type actions, if provided, the key to search for in the unsolicited messages.
command-detect-time	String	For command type actions, if provided, the time in seconds to detect for the command-unsol-key in the unsolicited messages. Also, the minimum time before a command response is checked for after the command is submitted for execution. If not provided, the default command-detect-time is 15 seconds when the command-unsol-key is specified or 10 seconds when the command-unsol-key is not specified.
workflow-definition-file	String	For workflow type actions, the workflow definition file.
workflow-variable-input-file	String	For workflow type actions, if provided, the workflow variable input file.
workflow-variables	Variable[]	For workflow type actions, if provided, the workflow variables. See Table 131 on page 187.
instructions	String	For instruction type actions, the instructions.
prompt-variables	PromptVariable[]	Prompt variables, for workflow type actions, if any are provided. At create time, there are prompts for the values. See Table 132 on page 187.
at-create-variables	String	Names of the at create variables, for workflow type actions, if any are provided. These are the only variables that are allowed on input variables for the do action operation.

Table 131. Variable structure

Field	Type
name	String
value	String
visibility	String. The value must be public or private.

Table 132. Prompt variable structure

Field	Type
name	String
value	String
abstract	String
description	String
error-message	String
label	String
max	String
min	String
milti-line	Boolean
must-be-choice	Boolean
choices	String List
places	String
regex	String
required	Boolean
type	String

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

The user issuing the request must be the owner of the software services instance, or, for catalog registry type objects, a domain administrator of the software services instance, or, for general registry type objects, the landlord.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `204 Normal` is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 133. HTTP error response codes for a update software services instance request

HTTP error status code	Description
`HTTP 400 Bad request`	Request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requestor user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance instance was not found; the software services instance does not exist.

Table 133. HTTP error response codes for a update software services instance request (continued)

HTTP error status code	Description
HTTP 409	The field cannot be updated for the registry type.

Response content

On successful completion, the response body contains nothing.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 134. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the PUT method is used to update a software services instance. The software services instance is uniquely identified by the software services instance key, which is represented by the following string value: `76963ea5-81a4-42d6-99d6-f3e19747cf61`.

```
PUT /zosmf/provisioning/rest/ 1.0/scr/ 76963ea5-81a4-42d6-99d6-f3e19747cf61
{
            "state":"provisioned"
}
```

Figure 80. Sample request to update a software services instance property

Update variables in a software services instance

You can use this operation to update variables in a software services instance.

HTTP method and URI path

```
PUT /zosmf/provisioning/rest/version/scr/object-id/variables
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance for which variables are to be updated.

Query parameters

None.

Description

This operation updates variables in a software services instance. If it already exists, the value and visibility are updated based on the values in the variable structure.

Request content

The request body contents must be a JSON object containing a variables field with the variables to be updated in the object. See Table 135.

Table 135. Request content for the update software services instance variables request

Field name	Type	Description
variables	Variable[]	The variables for the software, with the structure that is described in Table 136. The name field identifies the variable. If a variable in the variables array does not already exist in the software object it is added. If it does already exist the name and visibility are updated based on the values in the variable structure.

Table 136. Variable structure

Field	Type
name	String
value	String
visibility	String. The value must be public or private.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <*SAF-prefix*>.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

The user issuing the request must be the owner of the software services instance, or, for catalog registry type objects, a domain administrator of the software services instance, or, for general registry type objects, the landlord.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `204 Normal` is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 137. HTTP error response codes for a update software services instance request

HTTP error status code	Description
`HTTP 400 Bad request`	Request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requestor user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance instance was not found.

Response content

On successful completion, the response body contains nothing.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 138. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the PUT method is used to update variables for a software services instance. The software services instance is uniquely identified by the software services instance key, which is represented by the following string value: `76963ea5-81a4-42d6-99d6-f3e19747cf61`.

```
PUT  /zosmf/provisioning/rest/1.0/scr/76963ea5-81a4-42d6-99d6-f3e19747cf61/variables
Variable structure:
{
   "variables":[
      {"name":"var1","value":"val1","visibility":"public"},
      {"name":"var2","value":"val2","visibility":"public"}
   ]
}
```

Figure 81. Sample request to update variables for a software services instance

Delete a software services instance

The delete operation removes a software services instance from the software services registry.

HTTP method and URI path

```
DELETE /zosmf/provisioning/rest/<version>/scr/<object-id>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be deleted.

Query parameters

None.

Description

This operation removes a z/OSMF software services instance. The state of a software services instance must be one of the following:

- deprovisioned
- deprovisioning-failed
- provisioning-failed.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

The user issuing the request must be the owner of the software services instance.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `204 Normal` is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 139. HTTP error response codes for a delete software services instance request

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance was not found because it does not exist.
`HTTP 409 Request conflict`	The software services instance could not be removed because its state was not either deprovisioned, deprovisioning-failed, or provisioning-failed.

Response content

On successful completion, the response body contains nothing.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 140. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the DELTE method is used to delete a software services instance. The software services instance is uniquely identified by a key, which is the string value `76963ea5-81a4-42d6-99d6-f3e19747cf61`.

```
DELETE /zosmf/provisioning/rest/ 1.0/scr/76963ea5-81a4-42d6-99d6-f3e19747cf61
```

Figure 82. Sample request to delete a software services instance

Perform an action against a software services instance

You can use this operation to perform an action against a software services instance.

HTTP method and URI path

```
POST /zosmf/provisioning/rest/<version>/scr/<object-id>/actions/<action>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance.
- *<action>* identifies the action to be performed.

Query parameters

None.

Description

This operation performs an action against a software services instance.

On successful completion, HTTP status code `200` (Normal) is returned and the response body is provided, as described in "Response content" on page 194.

Request content

The request content is expected to contain a JSON object. See Table 141 for the fields.

Table 141. Request content for the perform action software services instance request

Field name	Type	Required or optional	Description
input-variables	input Variable[]	Optional	The input variables to be used by workflow-type actions. See Table 142.

Table 142. Input variable structure

Field	Type
name	String
value	String

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

The user issuing the request must be the owner of the software services instance, or, for catalog registry type objects, a domain administrator of the software services instance, or, for general registry type objects, the landlord.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (Normal) is returned and the response body is provided, as described in "Response content."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 143. HTTP error response codes for a do action request.

HTTP error status code	Description
HTTP 400 Eror	Bad request.
HTTP 401 Unauthorized	The requestor user ID is not authorized for this request.
HTTP 404 Not found	The specified software services instance was not found because it does not exist.

Response content

On successful completion, the response body contains a JSON object consisting of the response from the action.

Table 144. Resonse body for the do action request.

Field name	Type	Description
action-id	String	The ID of the action object that was created by running the action. The action ID is used on further requests to the action object.
action-uri	String	The URI of the new action object.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 145. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the POST method is used to perform an action for a software services instance.

```
POST /zosmf/provisioning/rest/1.0/scr/81963ea5-81a4-42d6-99d6-f3e19747cf61/actions/start
```

Figure 83. Sample request to perform an action against a software services instance variables

An example of the response is shown in the figures that follow.

```
{
"action-id":"65963ea5-81a4-42d6-99d6-f3e19748cf61",
"action-uri":
"/zosmf/provisioning/rest/1.0/scr/76963ea5-81a4-42d6-99d6-f3e19747cf61/actions/65963ea5-81a4-42d6-99d6-f3e19748cf61"
}
```

Figure 84. Sample response from a get software services instance variables request

Get the response for an action performed against a software services instance

You can use this operation to retrieve information about the response for an action that was performed against a software services instance.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scr/<object-id>/actions/<action-id>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be retrieved.
- *<action-id>* identifies the actions object to be retrieved.

Query parameters

None.

Description

This operation retrieves an action object that describes the response for an action that was performed against a software services instance.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 147 on page 197.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For catalog registry type objects, the user issuing the request must be at least one of the following:

- The owner of the software services instance
- A member of the tenant of the software services instance
- A domain administrator of the software services instance.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 147 on page 197.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 146. HTTP error response codes for a get software services instance contents request

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance was not found; the software services instance does not exist.

Response content

On successful completion, the response body is a JSON object that contains the retrieved data. Table 147 lists the fields in the JSON object.

Table 147. JSON object that is returned for a get actions request

Field	Type	Description
action-id	String	The action ID for the action object.
name	String	The name for the action.
type	String	Type of the action.
is-deprovision	String must be: • true • false	If true, the action is a deprovision action. Otherwise, the action is not a deprovision action.
state	String must be: • in-progress • submitted • responded • warning • complete • failed	The current state of the action. The values submitted, responded, and warning are valid only for command type actions. For the command action state of warning, see the command-response, command-sol-key-hit, and command-detection-status fields. Either no response was received, the command-sol-key-hit is false, or the command-detection-status is expired.
ran-at-time	String	The time the do action operation was done to create the action, in ISO8601 format
ran-by-user	String	The user ID that ran the do action operation that created the action
instructions	String	The instructions associated with the action, or null if no instructions are associated
command	String	The command associated with the action, or null if no command is associated
command-response	String	The solicited messages response from the command.
command-sol-key-hit	String: null, true, or false	If the command-sol-key was specified, indicates whether the command-sol-key was found in the solicited message response. If the command-sol-key-hit is false then the action state is set to warning.
command-detection-message	String	If the command-unsol-key was specified and it was found in the unsolicited messages from the command, the message containing the command-unsol-key.
command-detection-status	String: null, waiting, expired, or detected	If the command-unsol-key was specified, this is the status of whether the command-unsol-key was found in the unsolicited messages. If the command-detection-status is expired then the action state is set to warning.
workflow-key	String	The workflow key of the workflow associated with the action, or null if no workflow is associated
workflow-current-step-name	String	The current workflow step name of the workflow associated with the action, or null
workflow-message-id	String	The workflow message ID for the workflow associated with the action, or null
workflow-message-text	String	The workflow message text for the workflow associated with the action, or null
workflow-name	String	The workflow name for the workflow associated with the action, or null

Table 147. JSON object that is returned for a get actions request (continued)

workflow-status-name	String	The workflow status name for the workflow associated with the action, or null.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 148. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the GET method is used to retrieve the response for an action that was performed for a software services instance.

```
GET /zosmf/provisioning/rest/1.0/scr/b0d1806f-7d42-4b8d-ad4b-8b8747642cc3/actions/f5c4df98-f9fd-4fca-b1a5-e0d1b7d1f0d9
```

Figure 85. Sample request to get software services instance actions

The following is an example of the response.

```
{
   "name":"Instructions1",
   "state":"complete",
   "type":"instructions",
   "is-deprovision":"false",
   "command":null,
   "command-response":null,
   "command-sol-key-hit":null,
   "command-detection-status":null,
   "command-detection-message":null,
   "instructions":"These are the instructions for the Instruction1 action.",
   "action-id":"f5c4df98-f9fd-4fca-b1a5-e0d1b7d1f0d9",
   "ran-at-time":"2015-10-26T18:29:20.949Z",
   "ran-by-user":"ZOSMFAD",
   "workflow-current-step-name":null,
   "workflow-key":null,
   "workflow-message-id":null,
   "workflow-message-text":null,
   "workflow-name":null,
   "workflow-status-name":null
 }
```

Figure 86. Sample response for performed actions

List the responses for actions performed against a software services instance

You can use this operation to list the responses for actions that were performed against a software services instance.

HTTP method and URI path

```
GET /zosmf/provisioning/rest/<version>/scr/<object-id>/actions
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance for which actions are to be retrieved.

Query parameters

You can specify the following query parameter on this request. Objects matching all query parameters are returned.

type
Optional, specifies the type of the software.

name
Optional, regular expression, specifies the name of the action object.

state
Optional, specifies the current state of the action:

- in-progress
- submitted
- responded
- warning
- complete
- failed.

If you specify no query parameters, then all actions are returned.

Description

This operation lists the action objects for actions that were performed against a software services instance.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 151 on page 200.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

For catalog registry-type objects, the user issuing the request must be at least one of the following:

- The owner of the software object
- A member of the tenant of the software object
- A domain administrator of the software object.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 151.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 149. HTTP error response codes for a get software services instance contents request

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance was not found; the software services instance does not exist.

Response content

On successful completion, the response body is a JSON object that contains the retrieved data. See Table 150 and Table 151 lists the fields in the JSON object.

Table 150. JSON object that is returned for a list actions request

Field	Type	Description
scr-list-actions	Array of objects	Array of action objects.

Table 151. Fields for each action object

Field	Type	Description
action-id	String	The action ID for the action object.
name	String	The name for the action.
type	String	Type of the action.
state	String. Must be one of the following: • in-progress • submitted • responded • warning • complete • failed.	The current state of the action. The values submitted, responded, and warning are valid only for command type actions.
ran-at-time	String	The time the do action operation was done to create the action, in ISO8601 format
ran-by-user	String	The user ID that ran the do action operation that created the action

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 152. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.

Table 152. Response from a request failure (continued)

Field	Type	Description
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the GET method is used to retrieve the list of responses for actions that were performed against a software services instance.

```
GET /zosmf/provisioning/rest/1.0/scr/b0d1806f-7d42-4b8d-ad4b-8b8747642cc3/actions
```

Figure 87. Sample request to list performed actions

The following is an example of the response.

```
{
     "scr-list-actions":
     [
          {
            "name":"Instructions1",
            "state":"complete",
            "type":"instructions",
            "action-id":"f5c4df98-f9fd-4fca-b1a5-e0d1b7d1f0d9",
            "ran-at-time":"2015-10-26T18:29:20.949Z",
            "ran-by-user":"ZOSMFAD"
          },
          {
            "name":"deprovision",
            "state":"complete",
            "type":"workflow",
            "action-id":"ae3ec9cc-9be3-42b4-98f5-aa64934e31a3",
            "ran-at-time":"2015-10-27T14:34:27.186Z",
            "ran-by-user":"ZOSMFAD"
          }
      ]
}
```

Figure 88. Sample response from a list actions request

Delete the response for an action performed against a software services instance

The delete operation removes the response for an action that was performed against a software services instance.

HTTP method and URI path

```
DELETE /zosmf/provisioning/rest/<version>/scr/<object-id>/actions/<action-id>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the provisioning service. The following value is valid: `1.0`.
- *<object-id>* identifies the software services instance to be deleted.
- *<action-id>* identifies the action to be deleted.

Query parameters

None.

Description

This operation removes the response for an action that was performed against a software services instance.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: *<SAF-prefix>*.ZOSMF.PROVISIONING.SOFTWARE_SERVICES.

The user issuing the request must be the owner of the software services instance, or, for catalog registry type objects, a domain administrator of the software services instance, or, for general registry type objects, the landlord.

For more information, see "Authorization requirements" on page 158.

HTTP status codes

On successful completion, HTTP status code `204 Normal` is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 153. HTTP error response codes for a delete action request

HTTP error status code	Description
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.
`HTTP 404 Not found`	The specified software services instance was not found because it does not exist.

Response content

On successful completion, the response body contains nothing.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 154. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In the following example, the DELTE method is used to delete the response for an action that was performed against a software services instance.

```
DELETE /zosmf/provisioning/rest/1.0/scr/b0d1806f-7d42-4b8d-ad4b-8b8747642cc3/actions/f5c4df98-f9fd-4fca-b1a5-e0d1b7d1f0d9
```

Figure 89. Sample request to delete a response for a performed action

Software service instance name services

The software service instance name (SSIN) services are application programming interfaces (APIs), which are implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to create and manage software service instance names.

For information about cloud provisioning, including a description of the roles, see "Cloud provisioning services" on page 46.

Table 155 lists the operations that the SSIN services provide.

Table 155. SSIN services: operations summary

Operation name	HTTP method and URI path
"Create software service instance names" on page 206	`POST /zosmf/resource-mgmt/rest/<version>/ssin`
"List the software service instance names" on page 209	`GET /zosmf/resource-mgmt/rest/<version>/ssin`
"Create a variable name" on page 211	`POST /zosmf/resource-mgmt/rest/<version>/ssin/variable-name`
"Create unique variable names" on page 213	`POST /zosmf/resource-mgmt/rest/<version>/unique-variable-names`

Authorization requirements

Use of the SSIN services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <SAF-prefix>.ZOSMF.PROVISIONING.RESOURCE_MANAGEMENT.

Error response content

For the 4*nn* HTTP error status codes, additional diagnostic information beyond the HTTP status code is provided in the response body for the request. This information is provided in the form of a JSON object containing the following fields:

Table 156. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Error logging

Errors from the software services instance services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about

working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 Normal
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 201 Created
The request succeeded and resulted in the creation of an object.

HTTP 400 Bad request
The request contained incorrect parameters.

HTTP 401 Unauthorized
The request cannot be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both. Or, the client did not authenticate to z/OSMF by using a valid WWW-Authenticate header.

Create software service instance names

You can use this operation to create software service instance names (SSINs).

HTTP method and URI path

```
POST  /zosmf/resource-mgmt/rest/<version>/ssin
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software service instance name service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates SSINs. It uses the name-prefix in the resource definition profile as a basis for creating the names. An initial SSIN is created when a software instance is provisioned. A maximum of 8 generated SSINs may exist in the resource definition profile that the software instance is using. Allocation of SSINs for a provisioned software instance are released when the software instance is deprovisioned. The name-prefix in the resource definition profile must end with the special wildcard character, *.

For the properties that you can specify in the request body, a JSON object, see "Request content."

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the creation of a SSINs.

Request content

The request content is expected to contain a JSON object. Table 157 lists the fields in the JSON object.

Table 157. Request content for the create SSIN request

Field name	Type	Required or optional	Description
template-id	String	Required	The ID of the template.
domain-id	String	Required	The ID of the domain.
tenant-id	String	Required	The ID of the tenant.
registry-id	String	Required	The ID of the software instance registry.
quantity	String	Required	The number of names to be generated. The value must be 1-7.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <SAF-prefix>.ZOSMF.PROVISIONING.RESOURCE_MANAGEMENT.

For more information, see "Authorization requirements" on page 204.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content" on page 207.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 158. HTTP error response codes for a create SSIN request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.

Response content

On successful completion, the service returns a JSON object named ssin-list consisting of the names that were created. See Table 159.

Table 159. Response from a create SSIN request

Field	Type	Description
ssin-list	Array	Software service instance names. See Table 160.

Table 160. Fields in the ssin-list array

Field	Type	Description
ssin	String	Software service instance name.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 161. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In Figure 90, a request is submitted to create 2 SSINs.

```
POST /zosmf/resource-mgmt/rest/1.0/ssin
{
     "domain-id": "izu$0",
     "registry-id": "046c3cb2-7ef2-40a0-8b10-a34d8a23e5fc",
     "template-id": "9eb7df8a-284c-4550-a92e-8150bc6fe68f",
     "tenant-id": "izu$000",
     "quantity": "2"
 }
```

Figure 90. Sample request to create SSINs

The response body is as follows.

```
|  {
|     "ssin-list": [
|       {
|         "ssin": "INAME101"
|       },
|       {
|         "ssin": "INAME201"
|       }
|     ]
|   }

|
```

List the software service instance names

You can use this operation to list the software service instance names (SSINs).

HTTP method and URI path

```
GET  /zosmf/resource-mgmt/rest/<version>/ssin
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software service instance name service. The following value is valid: `1.0`.

Query parameters

You can specify the following query parameter on this request. Objects matching all query parameters are returned.

`name`
Name of the object for which SSINs should be obtained.

`registry-id`
Identifier of the registry for which SSINs should be obtained.

If you specify no query parameters, then all SSINs are returned.

Description

The list operation returns software service instance names based on the input query.

On successful completion, HTTP status code `200` (Normal) is returned, along with a response body.

Request content

None.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <SAF-prefix>.ZOSMF.PROVISIONING.RESOURCE_MANAGEMENT.

See "Authorization requirements" on page 204.

HTTP status codes

On successful completion, HTTP status code `200` (Normal) is returned and the response body is provided, as described in "Response content" on page 210.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 162. HTTP error response codes for a list SSIN request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.

Response content

On successful completion, the service returns a JSON object named ssin-list consisting of the names that were created. See Table 163.

Table 163. Response from a list SSINs request

Field	Type	Description
ssin-list	Array	Software service instance names. See Table 164.

Table 164. Fields in the ssin-list array

Field	Type	Description
ssin	String	Software service instance name.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 165. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

In Figure 91, a request is submitted to list the SSINs for name=INAME.*.

```
GET /zosmf/resource-mgmt/rest/1.0/ssin?name=INAME.*
```

Figure 91. Sample request to list SSINs

The response body is as follows.

```
{
   "ssin-list": [
     {
       "ssin": "INAME101"
     },
     {
       "ssin": "INAME201"
     }
   ]
 }
```

Create a variable name

You can use this operation to create a variable name.

HTTP method and URI path

```
POST  /zosmf/resource-mgmt/rest/<version>/ssin/variable-name
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software service instance name service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates a variable name based on the input variable prefix and the last 2 digits from the SSIN for the input registry ID. For the properties that you can specify, see "Request content."

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the creation of a new variable name. A response body is provided, as described in "Response content" on page 212.

Request content

The request content contains a JSON object. Table 166 lists the fields in the JSON object.

Table 166. Request content for the create variable name request

Field name	Type	Required or optional	Description
variable-prefix	String	Required	The prefix to use to create the variable name.
registry-id	String	Required	The ID of the software instance registry entry.

Authorization requirements

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <SAF-prefix>.ZOSMF.PROVISIONING.RESOURCE_MANAGEMENT.

See "Authorization requirements" on page 204.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content" on page 212.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 167. HTTP error response codes for a create variable request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained incorrect parameters.

Table 167. HTTP error response codes for a create variable request (continued)

HTTP error status code	Description
HTTP 401 Unauthorized	The requester user ID is not authorized for this request.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the variable name. Table 168 lists the fields in the JSON object.

Table 168. Response from a create variable name request

Field	Type	Description
name	String	Variable name.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 169. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

Figure 92 shows a request to create a variable name.

```
POST /zosmf/resource-mgmt/rest/1.0/ssin/variable-name

{
    "variable-prefix": "VAR",
    "registry-id": "3196202f-9a6c-4fdf-8dcd-e307e3ce2d5b"
 }
```

Figure 92. Sample request to create a variable name

The response body is as follows.

```
{
   "name": "VAR00"
 }
```

Create unique variable names

You can use this operation to create multiple unique variable names.

HTTP method and URI path

```
POST  /zosmf/resource-mgmt/rest/<version>/unique-variable-names
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF software service instance name service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates up to 50 unique variable names. For the properties that you can specify, see "Request content."

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the creation of unique variable names. A response body is provided, as described in "Response content" on page 214.

Request content

The request content contains a JSON object. Table 170 lists the fields in the JSON object.

Table 170. Request content for the create unique variable names request

Field name	Type	Required or optional	Description
prefix	String	Optional	The prefix to be used when creating the variable names. If a prefix is not specified on the request, one is supplied by the service.
quantity	String	Required	The number of names to be generated, 1-50.

Authorization requirements

None.

For more information, see "Authorization requirements" on page 204.

HTTP status codes

On successful completion, HTTP status code `201` (Created) is returned and the response body is provided, as described in "Response content" on page 214.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body that provides the reason code that is indicated and associated error message.

Table 171. HTTP error response codes for a create variable request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained incorrect parameters.
`HTTP 401 Unauthorized`	The requester user ID is not authorized for this request.

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the variable names. Table 172 lists the fields in the JSON object.

Table 172. Response from a create unique variable names request

Field	Type	Description
name-list	String Array	The variable names that were created.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 173. Response from a request failure

Field	Type	Description
httpStatus	Integer	HTTP status code.
requestMethod	String	HTTP request method.
requestUri	String	HTTP request URI.
messageID	String	Message identifier for the error.
messageText	String	Message text describing the error.
additionalInfo	String	Additional information describing the error.
debug	String	Debug information about for the error.

Example HTTP interaction

Figure 93 shows a request to create 5 unique variable names.

```
POST /zosmf/resource-mgmt/rest/1.0/unique-variable-name

{
     "prefix": "VAR",
     "quantity": "5"
 }
```

Figure 93. Sample request to create a variable name

The response body is as follows.

```
{
   "name-list": [
     "VAR1462458322735",
     "VAR1462458322737",
     "VAR1462458322739",
     "VAR1462458322741",
     "VAR1462458322743"
   ]
 }
```

Data persistence services

The data persistence services is an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. A set of REST services is provided for working with user-specific data and global application data, as described in this topic.

Table 174 lists the operations that the data persistence services provide.

Table 174. Operations provided through the data persistence services

Operation	HTTP method and URI path
"Persist user or application data" on page 217	PUT /zosmf/IzuUICommon/persistence/user/<*pluginId*>/<*taskId*>/<*resourcePath*> PUT /zosmf/IzuUICommon/persistence/app/<*pluginId*>/<*taskId*>/<*resourcePath*>
"Retrieve persisted user or application data" on page 220	GET /zosmf/IzuUICommon/persistence/user/<*pluginId*>/<*taskId*>/<*resourcePath*> GET /zosmf/IzuUICommon/persistence/app/<*pluginId*>/<*taskId*>/<*resourcePath*>
"Delete persisted user or application data" on page 223	DELETE /zosmf/IzuUICommon/persistence/user/<*pluginId*>/<*taskId*>/<*resourcePath*> DELETE /zosmf/IzuUICommon/persistence/app/<*pluginId*>/<*taskId*>/<*resourcePath*>

Required authorizations

The user must be logged in to z/OSMF, and must have READ access to the SAF profile that was registered for the plug-in and task making the request.

For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP request and response data

The JSON content type (`"Content-Type: application/json"`) is used for request and response data. The following JSON object is used by all data persistence services as input and output for the requested operations. The attributes provided in the JSON object depend on the requested operation.

```
{
 "value": "data-value",
 "version": "structure-version",
 "messages": "z/OSMF-messages",
 "update": true|false
}
```

where:

data-value
The value that will be added, updated, retrieved, or removed by the data persistence services. Any data type is supported including JSON objects, JSON arrays, and scalars. The value is required.

structure-version
Version of the data persistence services and the JSON object structure used for this request. The version sequence starts at 1.0.0, and is incremented only if the services or the JSON structure changes. The version the client supports is required as input to the request. The data persistence services is backward compatible for *n*-2 versions, and accepts requests for each version it supports. If the version specified by the client is not supported or if no version is specified, the service returns an error message.

z/OSMF-messages
z/OSMF messages received during the request. The *messages* attribute is included in the JSON

object only if an error occurred during the request. The message ID, message text, and stack trace are provided for each z/OSMF message received.

update
An optional input attribute, which indicates that the service is updating or replacing an existing JSON object. If you set the value to *true*, the service updates the key-value pairs you specified for the *value* attribute and preserves any other data persisted in the JSON object. You can set this attribute to *true* only when the data type is a JSON object or JSON array. If you omit this attribute or set it to *false*, the service deletes the existing JSON object and creates a new JSON object that contains only the key-value pairs you specified for the *value* attribute.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains a message that describes the error.

The following HTTP status codes are valid:

HTTP 200 OK
Success.

HTTP 400 Bad request
Request contained incorrect parameters.

HTTP 401 Unauthorized
Submitter of the request did not authenticate to z/OSMF or is not authorized to use the data persistence services.

HTTP 404 Bad URL
Target of the request (a URL) was not found.

HTTP 500 Internal server error
Programming error.

Error logging

Errors from the data persistence services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Persist user or application data

You can use this operation to persist data to be used by a specific user or application.

HTTP method and URI path

```
PUT /zosmf/IzuUICommon/persistence/user/<pluginId>/<taskId>/<resourcePath>
PUT /zosmf/IzuUICommon/persistence/app/<pluginId>/<taskId>/<resourcePath>
```

where:

- **zosmf/IzuUICommon/persistence** identifies the data persistence services.
- **user** indicates that the service will persist the data only for the user who is logged into z/OSMF when the service is invoked.
- **app** indicates that the service will persist the data globally for the application.
- **<pluginId>** is the unique identifier you assigned to the plug-in.
- **<taskId>** is the unique identifier you assigned to the task.
- **<resourcePath>** is the path in the JSON object to the attribute where you want the data to be stored. The persisted data is stored in a JSON object using a tree structure. To persist data, specify all the nodes or branches that must be traversed in the JSON structure to access that data. Use a forward slash (/) to separate each node or branch, and specify the nodes in the order in which they are listed in the structure.

  For example, to persist data for the *history* attribute shown in the sample JSON object in Figure 94, specify the following resource path: `/SETTINGS/history/`.

```
{
   "private": {
      "created": "2013-07-09T02:52:47.921Z",
      "majorv": 0,
      "minorv": 0,
      "modified": "2014-01-13T15:01:39.409Z"
   },
   "public": {
      "SETTINGS": {
         "authorization": {
            "auth": true
         },
         "history": {
            "acct": [
               "OMVS0803"
            ],
            "proc": [
               "CEANNKJ"
            ],
            "rsize": [
               "50000"
            ],
            "ugrp": [
               "ZOSMFGRP"
            ]
         },
         "trace": {
            "init": false,
            "task": false
         }
      }
   }
}
```

Figure 94. Sample JSON structure for persisted data

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that contains the value to be persisted and the version. For more details, see "Content type used for HTTP request and response data" on page 215.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 215.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes a JSON object that contains the current data after being modified. For more details, see "Content type used for HTTP request and response data" on page 332.

Example

To persist data that satisfies the following criteria, submit the request depicted in Figure 95:

- The data is for a task with the ID *MYTASK* that resides in plug-in *com.ibm.zoszmf.myapp*.
- The data is being persisted for the user who is currently logged into z/OSMF.
- The JSON object that contains the persistence data uses the structure provided in Figure 94 on page 217.
- The data to be persisted is updating the *rsize* attribute.

```
PUT /zosmf/IzuUICommon/persistence/user/com.ibm.zoszmf.myapp/MYTASK/SETTINGS/history/ HTTP/1.1
Host: zosmf1.yourco.com
Accept: application/json
Content-Type: application/json

{
  "version" : "1.0.0",
  "value" : {"history":{"rsize":["40000"]}},
  "update" : true
}
```

Figure 95. Sample request to persist user-specific data

A sample response is shown in Figure 96 on page 219.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Content-Type: application/json

{
  "version" : "1.0.0",
  "value" :
  {
    "authorization":{"auth":true},
    "history":{"ugrp":["ZOSMFGRP"],"acct":["OMVS0803"],"rsize":["40000"],"proc":["CEANNKJ"]},
    "trace":{"init":false,"task":false}
  }
}
```

Figure 96. Sample response from a request to persist user-specific data

Retrieve persisted user or application data

You can use this operation to retrieve data that is persisted for a specific user or application.

HTTP method and URI path

```
GET /zosmf/IzuUICommon/persistence/user/<pluginId>/<taskId>/<resourcePath>
GET /zosmf/IzuUICommon/persistence/app/<pluginId>/<taskId>/<resourcePath>
```

where:

- **zosmf/IzuUICommon/persistence** identifies the data persistence services.
- **user** indicates that the service will retrieve the data that has been persisted for the user who is logged into z/OSMF when the service is invoked.
- **app** indicates that the service will retrieve the data that has been persisted globally for the application.
- **<pluginId>** is the unique identifier you assigned to the plug-in.
- **<taskId>** is the unique identifier you assigned to the task.
- **<resourcePath>** is the path in the JSON object to the persisted data. The persisted data is stored in a JSON object using a tree structure. To retrieve persisted data, specify all the nodes or branches that must be traversed in the JSON structure to access that data. Use a forward slash (/) to separate each node or branch, and specify the nodes in the order in which they are listed in the structure.

  For example, to retrieve the data persisted for the *history* attribute shown in the sample JSON object in Figure 97, specify the following resource path: `/SETTINGS/history/`. In which case, the value for the *acct*, *proc*, *rsize*, and *ugrp* attributes will be retrieved. To retrieve the value for only the *rsize* attribute, specify the following resource path: `/SETTINGS/history/rsize/`.

```
{
   "private": {
      "created": "2013-07-09T02:52:47.921Z",
      "majorv": 0,
      "minorv": 0,
      "modified": "2014-01-13T15:01:39.409Z"
   },
   "public": {
      "SETTINGS": {
         "authorization": {
            "auth": true
         },
         "history": {
            "acct": [
               "OMVS0803"
            ],
            "proc": [
               "CEANNKJ"
            ],
            "rsize": [
               "50000"
            ],
            "ugrp": [
               "ZOSMFGRP"
            ]
         },
         "trace": {
            "init": false,
            "task": false
         }
      }
   }
}
```

Figure 97. Sample JSON structure for persisted data

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 215.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes a JSON object that contains the retrieved data. For more details, see "Content type used for HTTP request and response data" on page 332.

Example

To retrieve the persisted data that satisfies the following criteria, submit the request depicted in Figure 98:

- The data was persisted for a task with the ID *MYTASK* that resides in plug-in *com.ibm.zoszmf.myapp*.
- The data was persisted for the user who is currently logged into z/OSMF.
- The JSON object that contains the data uses the structure provided in Figure 97 on page 220.
- The data persisted for the *SETTINGS* attribute is to be retrieved.

```
GET /zosmf/IzuUICommon/persistence/user/com.ibm.zoszmf.myapp/MYTASK/SETTINGS/ HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 98. Sample request to retrieve persisted data

A sample response is shown in Figure 99 on page 222.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2014 05:39:28 +0000GMT
Connection: close

{
 "value":{
  "authorization":{"auth":true},
  "history":{"ugrp":["ZOSMFGRP"],"acct":["OMVS0803"],"rsize":["50000"],"proc":["CEANNKJ"]},
  "trace":{"init":false,"task":false}
 }
 "version":"1.0.0"
}
```

Figure 99. Sample response from a request to retrieve persisted data

Delete persisted user or application data

You can use this operation to remove data that is persisted for a specific user or application.

HTTP method and URI path

```
DELETE /zosmf/IzuUICommon/persistence/user/<pluginId>/<taskId>/<resourcePath>
DELETE /zosmf/IzuUICommon/persistence/app/<pluginId>/<taskId>/<resourcePath>
```

where:

- **zosmf/IzuUICommon/persistence** identifies the data persistence services.
- **user** indicates that the service will delete data that has been persisted for the user who is logged into z/OSMF when the service is invoked.
- **app** indicates that the service will delete data that has been persisted globally for the application.
- **<pluginId>** is the unique identifier you assigned to the plug-in.
- **<taskId>** is the unique identifier you assigned to the task.
- **<resourcePath>** is the path in the JSON object to the data to be deleted. The persisted data is stored in a JSON object using a tree structure. To delete persisted data, specify all the nodes or branches that must be traversed in the JSON structure to access that data. Use a forward slash (/) to separate each node or branch, and specify the nodes in the order in which they are listed in the structure.

  For example, to delete the data persisted for the *history* attribute shown in the sample JSON object in Figure 100, specify the following resource path: `/SETTINGS/history/`. In which case, the value for the *acct*, *proc*, *rsize*, and *ugrp* attributes will be deleted. To delete the value for only the *rsize* attribute, specify the following resource path: `/SETTINGS/history/rsize/`.

```
{
   "private": {
      "created": "2013-07-09T02:52:47.921Z",
      "majorv": 0,
      "minorv": 0,
      "modified": "2014-01-13T15:01:39.409Z"
   },
   "public": {
      "SETTINGS": {
         "authorization": {
            "auth": true
         },
         "history": {
            "acct": [
               "OMVS0803"
            ],
            "proc": [
               "CEANNKJ"
            ],
            "rsize": [
               "50000"
            ],
            "ugrp": [
               "ZOSMFGRP"
            ]
         },
         "trace": {
            "init": false,
            "task": false
         }
      }
   }
}
```

Figure 100. Sample JSON structure for persisted data

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 215.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes the updated JSON object. For more details, see "Content type used for HTTP request and response data" on page 332.

Example

To delete the persisted data that satisfies the following criteria, submit the request depicted in Figure 101:

- The data was persisted for a task with the ID *MYTASK* that resides in plug-in *com.ibm.zoszmf.myapp*.
- The data was persisted for the user who is currently logged into z/OSMF.
- The JSON object that contains the data uses the structure provided in Figure 100 on page 223.
- The data persisted for the *history* attribute is to be deleted.

```
DELETE /zosmf/IzuUICommon/persistence/user/com.ibm.zoszmf.myapp/MYTASK/SETTINGS/history HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 101. Sample request to delete persisted data

A sample response is shown in Figure 102 on page 225.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{
"version" : "1.0.0",
"value":{
  "authorization":{"auth":true},
  "history":null,
  "trace":{"init":false,"task":false}
 }
}
```

Figure 102. Sample response from a request to delete persisted data

Multisystem routing services

To communicate with and transfer data between systems within your enterprise, z/OSMF uses z/OSMF-to-z/OSMF communication. That is, a z/OSMF instance communicates with other z/OSMF instances to collect information from or about the systems in your enterprise. To enable this capability, each system in your enterprise must be accessible by a z/OSMF instance. Typically, this requires deploying one z/OSMF instance in each monoplex or sysplex in your enterprise.

Although your enterprise can have multiple active z/OSMF instances, it is recommended that you make one instance the primary. The *primary z/OSMF instance* is the instance that:

- Is the base for configuring the other z/OSMF instances in your enterprise.
- Generates the Lightweight Third Party Authentication (LTPA) key that is used for single sign-on (if single sign-on is enabled).
- Is used to perform all z/OS system management tasks in your enterprise, which ensures that the data for each z/OSMF task is centrally managed.
- Acts as the client for all hypertext transfer protocol (HTTP) requests and drives the transfer of files between z/OSMF instances.

You can select any z/OSMF instance that is at least z/OSMF V2R1 with APAR PI32148 to be the primary instance. The remaining z/OSMF instances are referred to as *secondary z/OSMF instances*.

Example

For example, suppose your installation is configured similar to the installation depicted in Figure 103 on page 227. The installation contains three sysplexes with a total of nine running systems. A z/OSMF instance is active in each sysplex, and your web browser is connected to the z/OSMF instance that is running on System 3 in Sysplex A. Thus, this z/OSMF instance is the primary instance and the z/OSMF instance running on System 6 in Sysplex B and System 9 in Sysplex C are the secondary instances.

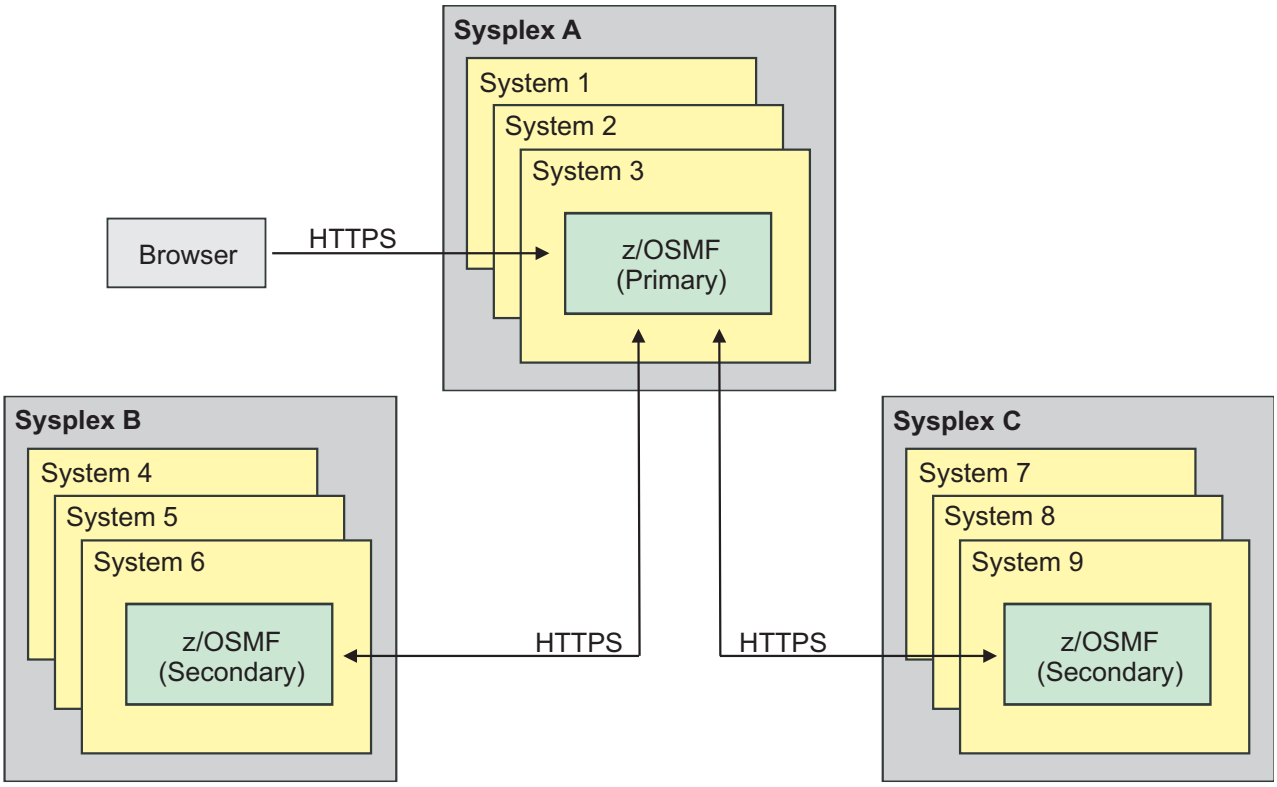

Figure 103. Example sysplex and system configuration

To obtain information from System 6 in Sysplex B, the following actions are performed:

1. A client application sends an HTTPS request to the primary z/OSMF instance.
2. The primary z/OSMF instance routes the HTTPS request to the secondary z/OSMF instance in Sysplex B.
3. The secondary z/OSMF instance processes the request and sends an HTTPS response to the primary z/OSMF instance.
4. The primary z/OSMF instance returns the HTTPS response to the browser.
5. The client application parses the response and extracts the appropriate information.

To route the request, the primary z/OSMF instance needs a system definition that specifies how to access the z/OSMF instance that is running on System 6 in Sysplex B and an HTTP proxy definition that specifies how to navigate the HTTP proxy server that is between the primary and secondary z/OSMF instances.

You can use the z/OSMF Systems task to add or modify the system and HTTP proxy definitions, and you can use the multisystem routing services to route the HTTPS request to the secondary z/OSMF instance and receive the HTTPS response. The remainder of this section describes the multisystem routing services. For information about the Systems task, see the z/OSMF online help.

Multisystem routing services overview

The multisystem routing services is an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. A set of REST services is provided for routing HTTPS requests to multiple systems in your enterprise. The multisystem routing services can route any HTTPS request that is supported by the z/OSMF REST services described in Chapter 1, “Using the z/OSMF REST services,” on page 1.

Table 175 lists the operations that the multisystem routing services provide.

Table 175. Operations provided through the multisystem routing services.

Operation	HTTP method and URI path
"Retrieve data from one or more systems" on page 231	GET /zosmf/gateway/system?content=<*http-content*> GET /zosmf/gateway/systems?content=<*http-content*> GET /zosmf/gateway/group?content=<*http-content*> GET /zosmf/gateway/sysplex?content=<*http-content*> GET /zosmf/gateway/cpc?content=<*http-content*>
"Update data for one or more systems" on page 240	POST /zosmf/gateway/system POST /zosmf/gateway/systems POST /zosmf/gateway/group POST /zosmf/gateway/sysplex POST /zosmf/gateway/cpc PUT /zosmf/gateway/system PUT /zosmf/gateway/systems PUT /zosmf/gateway/group PUT /zosmf/gateway/sysplex PUT /zosmf/gateway/cpc
"Delete data from one or more systems" on page 247	DELETE /zosmf/gateway/system?content=<*http-content*> DELETE /zosmf/gateway/systems?content=<*http-content*> DELETE /zosmf/gateway/group?content=<*http-content*> DELETE /zosmf/gateway/sysplex?content=<*http-content*> DELETE /zosmf/gateway/cpc?content=<*http-content*>
"Authenticate with a secondary z/OSMF instance" on page 254	POST /zosmf/gateway/logon
"Authenticate with an HTTP proxy server" on page 256	POST /zosmf/gateway/logon/proxy

Required authorizations

The user must be logged into z/OSMF. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP response data

The JSON content type (`"Content-Type: application/json"`) is used for HTTP response data. The following JSON object is used by all multisystem routing services for returning data and status about the requested operations. The attributes provided in the JSON object depend on the requested operation.

```
{
 "primaryAPIVersion":"primary-API-version",
 "systemsOutput":
   {
     "systemOutput":"system-output",
     "rc":"return-code",
     "error":{"msgid":"message-ID","msgtxt":"message-text"},
     "secondaryApiVersion":"secondary-API-version",
     "systemVersion":"
       {
         "zosNode":"zos-node",
```

```
        "zosVrm":"zos-level",
        "zosSysplex:"sysplex-name"
      }",
    "systemName":"system-name"
  },
 "numOfSystems":"total-systems"
}
```

where:

primary-API-version
Version of the multisystem routing services interface for the primary z/OSMF instance.

systemsOutput
Contains a separate response for each system to which the HTTPS request was sent. If the request was sent to multiple systems, the systemsOutput attribute contains an array of system responses.

system-output
Contains the response returned for a single system. A separate systemOutput attribute is included for each system to which the HTTPS request was sent.

return-code
Code returned by the system. The return code can be one of the following values:

OK Success.

HttpConnectionFailed
The HTTPS connection failed. Typically, this error occurs when the system hosting the secondary z/OSMF instance is unavailable, the z/OSMF instance is not running, or a network error has occurred.

HttpConnectionTimedOut
The HTTPS request did not complete in the time allotted.

CertificateError
The certificate for the secondary z/OSMF instance is not trusted.

LoginRequired
The Lightweight Third Party Authentication (LTPA) token is not valid or has expired. You must submit a separate HTTPS request to authenticate with the z/OSMF instance.

InvalidLogin
The login credentials for the z/OSMF instance are not valid.

ProxyLoginRequired
Authentication is required by the proxy server.

InvalidProxyLogin
The login credentials for the proxy server are not valid.

FailedWithMessage
The request was successful; however, an internal error occurred with the secondary z/OSMF instance.

UnexpectedFailure
An unexpected error occurred.

error If an error occurred with the request, the error attribute contains the message ID (msgid) and message text (msgtxt) for the message that was issued. Otherwise, this attribute is *null*.

secondary-API-version
Version of the multisystem routing services interface for the secondary z/OSMF instance.

systemVersion
Provides additional information about the system, as follows:

zosNode
JES2 multi-access spool (MAS) member name or JES3 complex member name that is assigned to the primary job entry subsystem (JES) that is running on the system.

zosVrm
Version, release, and modification level of the z/OS image installed on the system. The

level has the format *vv.rr.mm*, where *vv* is the version, *rr* is the release, and *mm* is the modification level. You can correlate the returned value as follows:

- 04.24.00 indicates z/OS V2R1

zosSysplex
Name of the sysplex where the z/OS system is a member. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.

system-name
Unique name assigned to the system definition.

total-systems
Number of systems to which an HTTPS request was sent.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains a message that describes the error.

The following HTTP status codes are valid:

HTTP 200 OK
Success.

HTTP 400 Bad request
Request contained incorrect parameters.

HTTP 401 Unauthorized
Submitter of the request did not authenticate with the primary or secondary z/OSMF instance, or is not authorized to use the z/OSMF REST service.

If the user ID required to authenticate with the primary and secondary z/OSMF instances are not the same, submit a separate HTTPS request to authenticate with the secondary z/OSMF instances.

HTTP 404 Bad URL
Target of the request (a URL) was not found.

HTTP 500 Internal server error
Programming error.

Error logging

Errors from the multisystem routing services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Retrieve data from one or more systems

You can use this operation to request that the primary z/OSMF instance submit an HTTPS request to retrieve data from one system, from a list of systems, or from all the systems in a group, sysplex, or central processor complex (CPC).

HTTP method and URI path

```
GET /zosmf/gateway/system?content=<http-content>
GET /zosmf/gateway/systems?content=<http-content>
GET /zosmf/gateway/group?content=<http-content>
GET /zosmf/gateway/sysplex?content=<http-content>
GET /zosmf/gateway/cpc?content=<http-content>
```

where:

- **zosmf/gateway** identifies the multisystem routing services.
- **system** informs the service that the request will be routed to only one system.
- **systems** informs the service that the request will be routed to a list of systems.
- **group** informs the service that the request will be routed to all of the systems in a group.
- **sysplex** informs the service that the request will be routed to all of the systems in a sysplex.
- **cpc** informs the service that the request will be routed to all of the systems in a CPC.
- **content=<*http-content*>** represents the parameters used to qualify the request. Table 176 lists the parameters that are supported for this request.

  Important: If the value for a parameter contains a number sign (#), encode the number sign as *%23*. Otherwise, everything following the number sign will be omitted from the request. For example, if the target is *System#1*, specify *System%231*.

Table 176. Supported input parameters for the multisystem routing services

Parameter	Required	Description
target	Yes	If the request is being sent to a system or a list of systems, the target is the nickname of the system. If the request is being sent to all the systems in a group, sysplex, or CPC, the target is the name of the group, sysplex, or CPC. The specified target must be defined in the Systems task. Otherwise, the request will fail.
resourcePath	Yes	Path to the z/OSMF REST service that will process the request. The resource path must be within the z/OSMF context. For example, to ping a TSO/E address space on the target system, you would use the TSO/E address space services to process the request. Therefore, you would specify the following resourcePath: `/tsoApp/ping/<servletKey>`, where *<servletKey>* identifies the TSO/E address space for the service to ping. When sending an HTTPS request to a list of systems, you can specify a different resource path and different parameters for each system included in the list. When sending an HTTPS request to all the systems in a group, sysplex, or CPC, you can specify only one resource path and one set of parameters, which will be used for all the systems in the specified group, sysplex, or CPC.
requestProperties	No	HTTP headers to be included in the HTTP request. Specify the HTTP headers as name and value pairs. If HTTP headers are omitted or are *null*, default values will be used, which are valid for most installations.

Table 176. Supported input parameters for the multisystem routing services (continued)

Parameter	Required	Description
timeout	No	Amount of time in milliseconds allowed to process a request. The value can range from 1 to 5601000 milliseconds. If omitted, the default value of 20000 milliseconds is used.
content	Yes if the HTTP method is POST or PUT.	Parameters or JSON object to include in the body of the HTTPS request that will be sent to the z/OSMF REST interface that will process the request.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 228.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 230.

The response also includes a JSON object that contains the requested information. For more details, see "Content type used for HTTP response data" on page 228.

Examples

To obtain sample HTTPS requests and responses for retrieving data from one system, from a list of systems, or from all the systems in a group, sysplex, or CPC, see the following sections:

- "Example 1: Retrieve data from one system" on page 233
- "Example 2: Retrieve data from a list of systems" on page 233
- "Example 3: Retrieve data from all the systems in a group" on page 235
- "Example 4: Retrieve data from all the systems in a sysplex" on page 237
- "Example 5: Retrieve data from all the systems in a CPC" on page 239

Example 1: Retrieve data from one system

To retrieve the handlers that are registered for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from system *sys057*, submit the following request:

```
GET /zosmf/gateway/system?content={"target":"sys057",
"resourcePath":"/izual/rest/handler?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP",
{"contentType":"application/json","charset":"UTF8"}} HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 104. Sample request to retrieve data from one system

A sample response is shown in Figure 105.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":
    {
      "systemOutput":
        {
          "error":null,
          "result":[
            {
              "id":"IBM.ZOSMF.IZU_IMPORT_HANDLER",
              "taskId":"IZUG_TASK_zOSMFImportManager",
              "enabled":true,
              "defaultHandler":false,
              "applId":"IzuImportManager",
              "type":"INTERNAL",
              "displayName":"Import Manager",
              "url":"/zosmf/IzuImportUtility/index.jsp",
              "eventTypeId":"IBM.ZOSMF.IMPORT_EXTERNAL_APP",
              "options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH"}
            }
          ],
        },
       "rc":"Ok",
       "secondaryApiVersion":1.0,
       "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
  "numOfSystems":1
}
```

Figure 105. Sample response from a request to retrieve data from one system

Example 2: Retrieve data from a list of systems

To retrieve the handlers that are registered for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from system *sys057* and for event type IBM.ZOSMF.VIEW_DATASET from system *sys060*, submit the following request:

```
GET /zosmf/gateway/systems?content=[{"target":"sys057",
"resourcePath":"/izual/rest/handler?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP",
{"contentType":"application/json","charset":"UTF8"}},{"target":"sys060",
"resourcePath":"/izual/rest/handler?eventTypeId=IBM.ZOSMF.VIEW_DATASET"}] HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 106. Sample request to retrieve data from a list of systems

A sample response is shown in Figure 107.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":[
            {
              "id":"IBM.ZOSMF.IZU_IMPORT_HANDLER",
              "taskId":"IZUG_TASK_zOSMFImportManager",
              "enabled":true,
              "defaultHandler":false,
              "applId":"IzuImportManager",
              "type":"INTERNAL",
              "displayName":"Import Manager",
              "url":"/zosmf/IzuImportUtility/index.jsp",
              "eventTypeId":"IBM.ZOSMF.IMPORT_EXTERNAL_APP",
              "options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH"}
            }
          ],
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":[
            {
              "id":"IBM.ISPF.ISR.EPDF.B",
              "enabled":true,
              "defaultHandler":false,
              "type":"EXTERNAL",
              "displayName":"ISR Browse Data Set",
              "url":
               "/zosmf/webispf/index.jsp?cmd=ISPSTART%20CMD(%25ISREPDF%20'%24dataSetName'%20B)%20NEWAPPL(ISR)",
              "eventTypeId":"IBM.ZOSMF.VIEW_DATASET",
              "options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH"}
            }
          ],
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY4","zosVrm":"04.24.00","zosSysplex":"PLEX4"}",
     "systemName":"sys060"
    }],
  "numOfSystems":2
}
```

Figure 107. Sample response from a request to retrieve data from a list of systems

Example 3: Retrieve data from all the systems in a group

To retrieve the handlers that are registered for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from all the systems in group *mygroup*, submit the following request:

```
GET /zosmf/gateway/group?content={"target":"mygroup",
"resourcePath":"/izual/rest/handler?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"} HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 108. Sample request to retrieve data from all the systems in a group

A sample response is shown in Figure 109 on page 236.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":[
            {
              "id":"IBM.ZOSMF.IZU_IMPORT_HANDLER",
              "taskId":"IZUG_TASK_zOSMFImportManager",
              "enabled":true,
              "defaultHandler":false,
              "applId":"IzuImportManager",
              "type":"INTERNAL",
              "displayName":"Import Manager",
              "url":"/zosmf/IzuImportUtility/index.jsp",
              "eventTypeId":"IBM.ZOSMF.IMPORT_EXTERNAL_APP",
              "options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH"}
            }
          ],
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG0000E",
            "msgtxt":"The HTTPS request to server "sys058" failed with return code
                     "LoginRequired" and HTTP response code "401"."
          },
          "result":null
        },
      "rc":"LoginRequired",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY2","zosVrm":"04.24.00","zosSysplex":"PLEX2"}",
      "systemName":"sys058"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG0000E",
            "msgtxt":"The HTTPS request to server "sys059" failed with return code
                     "HttpConnectionTimedOut" and HTTP response code "0"."
          },
          "result":null
        },
      "rc":"HttpConnectionTimedOut",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY3","zosVrm":"04.24.00","zosSysplex":"PLEX3"}",
      "systemName":"sys059"
    }],
  "numOfSystems":3
}
```

Figure 109. Sample response from a request to retrieve data from all the systems in a group

Example 4: Retrieve data from all the systems in a sysplex

To retrieve the handlers that are registered for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from all the systems in sysplex *PLEX1*, submit the following request:

```
GET /zosmf/gateway/sysplex?content={"target":"PLEX1",
"resourcePath":"/izual/rest/handler?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"} HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 110. Sample request to retrieve data from all the systems in a sysplex

A sample response is shown in Figure 111 on page 238.

```
HTTP/1.1 200 OK
Date: Thu, 15 Feb 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":[
            {
              "id":"IBM.ZOSMF.IZU_IMPORT_HANDLER",
              "taskId":"IZUG_TASK_zOSMFImportManager",
              "enabled":true,
              "defaultHandler":false,
              "applId":"IzuImportManager",
              "type":"INTERNAL",
              "displayName":"Import Manager",
              "url":"/zosmf/IzuImportUtility/index.jsp",
              "eventTypeId":"IBM.ZOSMF.IMPORT_EXTERNAL_APP",
              "options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH"}
            }
          ],
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG0000E",
            "msgtxt":"The HTTPS request to server "sys077" failed with return code
                     "LoginRequired" and HTTP response code "401"."
          },
          "result":null
        },
      "rc":"LoginRequired",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
      "systemName":"sys077"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG0000E",
            "msgtxt":"The HTTPS request to server "sys195" failed with return code
                     "HttpConnectionTimedOut" and HTTP response code "0"."
          },
          "result":null
        },
      "rc":"HttpConnectionTimedOut",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
      "systemName":"sys195"
    }],
  "numOfSystems":3
}
```

Figure 111. Sample response from a request to retrieve data from all the systems in a sysplex

Example 5: Retrieve data from all the systems in a CPC

To retrieve the handlers that are registered for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from all the systems in CPC *CPC1*, submit the following request:

```
GET /zosmf/gateway/cpc?content={"target":"CPC1",
"resourcePath":"/izual/rest/handler?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"} HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 112. Sample request to retrieve data from all the systems in a CPC

A sample response is shown in Figure 113.

```
HTTP/1.1 200 OK
Date: Thu, 15 Feb 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":[
            {
              "id":"IBM.ZOSMF.IZU_IMPORT_HANDLER",
              "taskId":"IZUG_TASK_zOSMFImportManager",
              "enabled":true,
              "defaultHandler":false,
              "applId":"IzuImportManager",
              "type":"INTERNAL",
              "displayName":"Import Manager",
              "url":"/zosmf/IzuImportUtility/index.jsp",
              "eventTypeId":"IBM.ZOSMF.IMPORT_EXTERNAL_APP",
              "options":{"CONTEXT_SUPPORT":"OPT_CONTEXT_SUPPORT_LAUNCH_AND_SWITCH"}
            }
          ],
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY5","zosVrm":"04.24.00","zosSysplex":"PLEX5"}",
      "systemName":"sys289"
    }],
  "numOfSystems":2
}
```

Figure 113. Sample response from a request to retrieve data from all the systems in a CPC

Update data for one or more systems

You can use this operation to request that the primary z/OSMF instance submit an HTTPS request to update data for one system, for a list of systems, or for all the systems in a group, sysplex, or central processor complex (CPC).

HTTP method and URI path

```
POST /zosmf/gateway/system
POST /zosmf/gateway/systems
POST /zosmf/gateway/group
POST /zosmf/gateway/sysplex
POST /zosmf/gateway/cpc
PUT /zosmf/gateway/system
PUT /zosmf/gateway/systems
PUT /zosmf/gateway/group
PUT /zosmf/gateway/sysplex
PUT /zosmf/gateway/cpc
```

where:

- **zosmf/gateway** identifies the multisystem routing services.
- **system** informs the service that the request will be routed to only one system.
- **systems** informs the service that the request will be routed to a list of systems.
- **group** informs the service that the request will be routed to all of the systems in a group.
- **sysplex** informs the service that the request will be routed to all of the systems in a sysplex.
- **cpc** informs the service that the request will be routed to all of the systems in a CPC.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object or JSON object stream that describes the objects to be created, updated, or modified for each system. Table 177 lists the supported parameters.

Table 177. Supported input parameters for the multisystem routing services

Parameter	Required	Description
target	Yes	If the request is being sent to a system or a list of systems, the target is the nickname of the system. If the request is being sent to all the systems in a group, sysplex, or CPC, the target is the name of the group, sysplex, or CPC. The specified target must be defined in the Systems task. Otherwise, the request will fail.

Table 177. Supported input parameters for the multisystem routing services (continued)

Parameter	Required	Description
resourcePath	Yes	Path to the z/OSMF REST service that will process the request. The resource path must be within the z/OSMF context. For example, to ping a TSO/E address space on the target system, you would use the TSO/E address space services to process the request. Therefore, you would specify the following resourcePath: `/tsoApp/ping/<servletKey>`, where *<servletKey>* identifies the TSO/E address space for the service to ping. When sending an HTTPS request to a list of systems, you can specify a different resource path and different parameters for each system included in the list. When sending an HTTPS request to all the systems in a group, sysplex, or CPC, you can specify only one resource path and one set of parameters, which will be used for all the systems in the specified group, sysplex, or CPC.
requestProperties	No	HTTP headers to be included in the HTTP request. Specify the HTTP headers as name and value pairs. If HTTP headers are omitted or are *null*, default values will be used, which are valid for most installations.
timeout	No	Amount of time in milliseconds allowed to process a request. The value can range from 1 to 5601000 milliseconds. If omitted, the default value of 20000 milliseconds is used.
content	Yes if the HTTP method is POST or PUT.	Parameters or JSON object to include in the body of the HTTPS request that will be sent to the z/OSMF REST interface that will process the request.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 228.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 230.

The response also includes a JSON object that contains the requested information. For more details, see "Content type used for HTTP response data" on page 228.

Examples

To obtain sample HTTPS requests and responses for updating data for one system, for a list of systems, or for all the systems in a group, sysplex, or CPC, see the following sections:

- "Example 1: Update data for one system" on page 242
- "Example 2: Update data for a list of systems" on page 242
- "Example 3: Update data for all the systems in a group" on page 243
- "Example 4: Update data for all the systems in a sysplex" on page 244
- "Example 5: Update data for all the systems in a CPC" on page 245

Example 1: Update data for one system

To create event type IBM.ZOSMF.VIEW_JOB_STATUS on system *sys057*, submit the following request:

```
POST /zosmf/gateway/system HTTP/1.1
Host: zosmf1.yourco.com

{"target":"sys057","resourcePath":"/izual/rest/eventtype","content":
{"id":"IBM.ZOSMF.VIEW_JOB_STATUS","displayName":"View Job Status",
"desc":"View the status of a job.","owner":"SDSF","params":{"jobName":
"Name of the job for which to view status."}}}
```

Figure 114. Sample request to update data for one system

A sample response is shown in Figure 115.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
  "numOfSystems":1
}
```

Figure 115. Sample response from a request to update data for one system

Example 2: Update data for a list of systems

To create event type IBM.ZOSMF.VIEW_JOB_STATUS on system *sys057* and event type IBM.ZOSMF.VIEW_WLM_STATUS on system *sys060,* submit the following request:

```
POST /zosmf/gateway/systems HTTP/1.1
Host: zosmf1.yourco.com

[{"target":"sys057","resourcePath":"/izual/rest/eventtype","content":
{"id":"IBM.ZOSMF.VIEW_JOB_STATUS","displayName":"View Job Status",
"desc":"View the status of a job.","owner":"SDSF","params":{"jobName":
"Name of the job for which to view status."}}},
{"target":"sys060","resourcePath":"/izual/rest/eventtype",
"content":{"id"="IBM.ZOSMF.VIEW_WLM_STATUS","displayName":"View WLM Status",
"desc":"View the status of WLM.","owner":"IBM","params":{"sysplex:"Name of the sysplex."}}}]
```

Figure 116. Sample request to update data for a list of systems

A sample response is shown in Figure 117 on page 243.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY4","zosVrm":"04.24.00","zosSysplex":"PLEX4"}",
     "systemName":"sys060"
    }],
  "numOfSystems":2
}
```

Figure 117. Sample response from a request to update data for a list of systems

Example 3: Update data for all the systems in a group

To create event type IBM.ZOSMF.VIEW_JOB_STATUS for all the systems in group *mygroup,* submit the following request:

```
POST /zosmf/gateway/group HTTP/1.1
Host: zosmf1.yourco.com

{"target":"mygroup","resourcePath":"/izual/rest/eventtype","content":
{"id":"IBM.ZOSMF.VIEW_JOB_STATUS","displayName":"View Job Status",
"desc":"View the status of a job.","owner":"SDSF","params":{"jobName":
"Name of the job for which to view status."}}}
```

Figure 118. Sample request to update data for all the systems in a group

A sample response is shown in Figure 119 on page 244.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY2","zosVrm":"04.24.00","zosSysplex":"PLEX2"}",
      "systemName":"sys058"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY3","zosVrm":"04.24.00","zosSysplex":"PLEX3"}",
      "systemName":"sys059"
    }],
  "numOfSystems":3
}
```

Figure 119. Sample response from a request to update data for all the systems in a group

Example 4: Update data for all the systems in a sysplex

To create event type IBM.ZOSMF.VIEW_JOB_STATUS for all the systems in sysplex *PLEX1,* submit the following request:

```
POST /zosmf/gateway/sysplex HTTP/1.1
Host: zosmf1.yourco.com

{"target":"PLEX1","resourcePath":"/izual/rest/eventtype","content":
{"id":"IBM.ZOSMF.VIEW_JOB_STATUS","displayName":"View Job Status",
"desc":"View the status of a job.","owner":"SDSF","params":{"jobName":
"Name of the job for which to view status."}}}
```

Figure 120. Sample request to update data for all the systems in a sysplex

A sample response is shown in Figure 121 on page 245.

```
HTTP/1.1 200 OK
Date: Thu, 15 Feb 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
      "systemName":"sys077"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
      "systemName":"sys195"
    }],
  "numOfSystems":3
}
```

Figure 121. Sample response from a request to update data for all the systems in a sysplex

Example 5: Update data for all the systems in a CPC

To create event type IBM.ZOSMF.VIEW_JOB_STATUS for all the systems in CPC *CPC1,* submit the following request:

```
POST /zosmf/gateway/cpc HTTP/1.1
Host: zosmf1.yourco.com

{"target":"CPC1","resourcePath":"/izual/rest/eventtype","content":
{"id":"IBM.ZOSMF.VIEW_JOB_STATUS","displayName":"View Job Status",
"desc":"View the status of a job.","owner":"SDSF","params":{"jobName":
"Name of the job for which to view status."}}}
```

Figure 122. Sample request to update data for all the systems in a CPC

A sample response is shown in Figure 123 on page 246.

```
HTTP/1.1 200 OK
Date: Thu, 15 Feb 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null,
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY5","zosVrm":"04.24.00","zosSysplex":"PLEX5"}",
      "systemName":"sys289"
    }],
  "numOfSystems":2
}
```

Figure 123. Sample response from a request to update data for all the systems in a CPC

Delete data from one or more systems

You can use this operation to request that the primary z/OSMF instance submit an HTTPS request to delete data from one system, from a list of systems, or from all the systems in a group, sysplex, or central processor complex (CPC).

HTTP method and URI path

```
DELETE /zosmf/gateway/system?content=<http-content>
DELETE /zosmf/gateway/systems?content=<http-content>
DELETE /zosmf/gateway/group?content=<http-content>
DELETE /zosmf/gateway/sysplex?content=<http-content>
DELETE /zosmf/gateway/cpc?content=<http-content>
```

where:

- **zosmf/gateway** identifies the multisystem routing services.
- **system** informs the service that the request will be routed to only one system.
- **systems** informs the service that the request will be routed to a list of systems.
- **group** informs the service that the request will be routed to all of the systems in a group.
- **sysplex** informs the service that the request will be routed to all of the systems in a sysplex.
- **cpc** informs the service that the request will be routed to all of the systems in a CPC.
- **content=<*http-content*>** represents the parameters used to qualify the request. Table 178 lists the parameters that are supported for this request.

  Important: If the value for a parameter contains a number sign (#), encode the number sign as *%23*. Otherwise, everything following the number sign will be omitted from the request. For example, if the target is *System#1*, specify *System%231*.

Table 178. Supported input parameters for the multisystem routing services

Parameter	Required	Description
target	Yes	If the request is being sent to a system or a list of systems, the target is the nickname of the system. If the request is being sent to all the systems in a group, sysplex, or CPC, the target is the name of the group, sysplex, or CPC. The specified target must be defined in the Systems task. Otherwise, the request will fail.
resourcePath	Yes	Path to the z/OSMF REST service that will process the request. The resource path must be within the z/OSMF context. For example, to ping a TSO/E address space on the target system, you would use the TSO/E address space services to process the request. Therefore, you would specify the following resourcePath: `/tsoApp/ping/<servletKey>`, where *<servletKey>* identifies the TSO/E address space for the service to ping. When sending an HTTPS request to a list of systems, you can specify a different resource path and different parameters for each system included in the list. When sending an HTTPS request to all the systems in a group, sysplex, or CPC, you can specify only one resource path and one set of parameters, which will be used for all the systems in the specified group, sysplex, or CPC.
requestProperties	No	HTTP headers to be included in the HTTP request. Specify the HTTP headers as name and value pairs. If HTTP headers are omitted or are *null*, default values will be used, which are valid for most installations.

Table 178. Supported input parameters for the multisystem routing services (continued)

Parameter	Required	Description
timeout	No	Amount of time in milliseconds allowed to process a request. The value can range from 1 to 5601000 milliseconds. If omitted, the default value of 20000 milliseconds is used.
content	Yes if the HTTP method is POST or PUT.	Parameters or JSON object to include in the body of the HTTPS request that will be sent to the z/OSMF REST interface that will process the request.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 228.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 230.

The response also includes a JSON object that contains the requested information. For more details, see "Content type used for HTTP response data" on page 228.

Examples

To obtain sample HTTPS requests and responses for deleting data from one system, from a list of systems, or from all the systems in a group, sysplex, or CPC, see the following sections:

- "Example 1: Delete data from one system" on page 249
- "Example 2: Delete data from a list of systems" on page 249
- "Example 3: Delete data from all the systems in a group" on page 250
- "Example 4: Delete data from all the systems in a sysplex" on page 251
- "Example 5: Delete data from all the systems in a CPC" on page 252

Example 1: Delete data from one system

To remove handler IBM.ZOSMF.IZU_IMPORT_HANDLER for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from system *sys057*, submit the following request:

```
DELETE /zosmf/gateway/system?content={"target":"sys057",
"resourcePath":"/izual/rest/handler/IBM.ZOSMF.IZU_IMPORT_HANDLER?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"}
HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 124. Sample request to delete data from one system

A sample response is shown in Figure 125.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
  "numOfSystems":1
}
```

Figure 125. Sample response from a request to delete data from one system

Example 2: Delete data from a list of systems

To remove handler IBM.ZOSMF.IZU_IMPORT_HANDLER for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from system *sys057* and to remove event type IBM.ZOSMF.VIEW_DATASET from system *sys060*, submit the following request:

```
DELETE /zosmf/gateway/systems?content=[{"target":"sys057",
"resourcePath":"/izual/rest/handler/IBM.ZOSMF.IZU_IMPORT_HANDLER?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"},
{"target":"sys060","resourcePath":"/izual/rest/eventtype/IBM.ZOSMF.VIEW_DATASET"}] HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 126. Sample request to delete data from a list of systems

A sample response is shown in Figure 127 on page 250.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG698E",
            "msgtxt":"The request could not be completed because 1 handlers are registered for
                      event type "IBM.ZOSMF.VIEW_DATASET"."
          },
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY4","zosVrm":"04.24.00","zosSysplex":"PLEX4"}",
     "systemName":"sys060"
    }],
  "numOfSystems":2
}
```

Figure 127. Sample response from a request to delete data from a list of systems

Example 3: Delete data from all the systems in a group

To remove handler IBM.ZOSMF.IZU_IMPORT_HANDLER for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from all the systems in group *mygroup*, submit the following request:

```
DELETE /zosmf/gateway/group?content={"target":"mygroup",
"resourcePath":"/izual/rest/handler/IBM.ZOSMF.IZU_IMPORT_HANDLER?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"}
HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 128. Sample request to delete data from all the systems in a group

A sample response is shown in Figure 129 on page 251.

```
HTTP/1.1 200 OK
Date: Fri, 16 Jan 2015 04:13:56 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY2","zosVrm":"04.24.00","zosSysplex":"PLEX2"}",
      "systemName":"sys058"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG0000E",
            "msgtxt":"The HTTPS request to server "sys059" failed with return code
                     "HttpConnectionTimedOut" and HTTP response code "0"."
          },
          "result":null
        },
      "rc":"HttpConnectionTimedOut",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY3","zosVrm":"04.24.00","zosSysplex":"PLEX3"}",
      "systemName":"sys059"
    }],
  "numOfSystems":3
}
```

Figure 129. Sample response from a request to delete data from all the systems in a group

Example 4: Delete data from all the systems in a sysplex

To remove handler IBM.ZOSMF.IZU_IMPORT_HANDLER for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from all the systems in sysplex *PLEX1,* submit the following request:

```
DELETE /zosmf/gateway/sysplex?content={"target":"PLEX1",
"resourcePath":"/izual/rest/handler/IBM.ZOSMF.IZU_IMPORT_HANDLER?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"}
HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 130. Sample request to delete data from all the systems in a sysplex

A sample response is shown in Figure 131 on page 252.

```
HTTP/1.1 200 OK
Date: Fri, 16 Feb 2015 04:13:56 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
      "systemName":"sys077"
    },
    {
      "systemOutput":
        {
          "error":
          {
            "msgid":"IZUG0000E",
            "msgtxt":"The HTTPS request to server "sys195" failed with return code
                     "HttpConnectionTimedOut" and HTTP response code "0"."
          },
          "result":null
        },
      "rc":"HttpConnectionTimedOut",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
      "systemName":"sys195"
    }],
  "numOfSystems":3
}
```

Figure 131. Sample response from a request to delete data from all the systems in a sysplex

Example 5: Delete data from all the systems in a CPC

To remove handler IBM.ZOSMF.IZU_IMPORT_HANDLER for event type IBM.ZOSMF.IMPORT_EXTERNAL_APP from all the systems in CPC *CPC1,* submit the following request:

```
DELETE /zosmf/gateway/cpc?content={"target":"CPC1",
"resourcePath":"/izual/rest/handler/IBM.ZOSMF.IZU_IMPORT_HANDLER?eventTypeId=IBM.ZOSMF.IMPORT_EXTERNAL_APP"}
HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 132. Sample request to delete data from all the systems in a CPC

A sample response is shown in Figure 133 on page 253.

```
HTTP/1.1 200 OK
Date: Fri, 16 Feb 2015 04:13:56 +0000GMT
Connection: close

{
  "primaryAPIVersion":1.0,
  "systemsOutput":[
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY1","zosVrm":"04.24.00","zosSysplex":"PLEX1"}",
     "systemName":"sys057"
    },
    {
      "systemOutput":
        {
          "error":null,
          "result":null
        },
      "rc":"Ok",
      "secondaryApiVersion":1.0,
      "systemVersion":"{"zosNode":"SY5","zosVrm":"04.24.00","zosSysplex":"PLEX5"}",
      "systemName":"sys289"
    }],
  "numOfSystems":2
}
```

Figure 133. Sample response from a request to delete data from all the systems in a CPC

Authenticate with a secondary z/OSMF instance

You can use this operation to request that the primary z/OSMF instance submit an HTTPS request to authenticate with a secondary z/OSMF instance.

HTTP method and URI path

```
POST /zosmf/gateway/logon
```

where:

- **zosmf/gateway** identifies the multisystem routing services.
- **logon** informs the service that the request is to authenticate with a system.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include the following JSON object:

```
{
  "userid":"user-ID",
  "password":"password",
  "systemName":"system-name"
}
```

where:

user-ID
: z/OS user ID that allows the user to access the specified system. The user ID is the same user ID that is specified in your installation's z/OS security management facility (for example, RACF). The user ID is required.

password
: Password or pass phrase associated with the z/OS user ID. The password is required.

system-name
: Unique name assigned to the system definition.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 228.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 230.

The response also includes a JSON object that indicates whether the request was successful. If the logon request is successful, the timeout for the Lightweight Third Party Authentication (LTPA) token is returned, as depicted in Figure 134.

```
{"timeout":7564710}
```

Figure 134. Successful response when authenticating with a system

If the logon request is unsuccessful, the JSON object contains an error message, as depicted in Figure 135.

```
{"error":true,"errMsg":"IZUG410E: The user ID, password, or pass phrase is not valid.
Enter the correct values for your security management product."}
```

Figure 135. Response when the authentication request fails

Example

To authenticate with system *sys057*, submit the following request:

```
POST /zosmf/gateway/logon HTTP/1.1
Host: zosmf1.yourco.com

{"userid":"claire","password":"abc123","systemName":"sys057"}
```

Figure 136. Sample request to authenticate with a system

Authenticate with an HTTP proxy server

You can use this operation to authenticate with the HTTP proxy server that the primary z/OSMF instance is required to navigate to communicate with a secondary z/OSMF instance.

HTTP method and URI path

```
POST /zosmf/gateway/logon/proxy
```

where:

- **zosmf/gateway** identifies the multisystem routing services.
- **logon/proxy** informs the service that the request is to authenticate with an HTTP proxy server.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include the following JSON object:

```
{
  "proxyUserId":"proxy-user-ID",
  "proxyPassword":"proxy-password",
  "systemName":"system-name"
}
```

where:

proxy-user-ID
: User ID that allows the user to access the HTTP proxy server at your enterprise. The user ID is required.

proxy-password
: Password or pass phrase associated with the proxy user ID. The password is required.

system-name
: Unique name assigned to the system definition that specifies the URL for accessing the secondary z/OSMF instance.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 228.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 230.

The response also includes a JSON object that indicates whether the request was successful. If the logon request is successful, *null* values are returned for the *result* and *error* attributes, as depicted in Figure 137.

```
{"result":null,"error":null}
```

Figure 137. Successful response when authenticating with an HTTP proxy server

If the logon request is unsuccessful, the JSON object contains an error message, as depicted in Figure 138. For a description of each attribute, see "Content type used for HTTP response data" on page 228.

```
{
 "primaryAPIVersion":1.0,
 "systemsOutput":null,
 "error":{
   "msgid":"IZUG476E",
   "msgtxt":"The HTTP request to the secondary z/OSMF instance "sys057"
   failed with error type "InvalidProxyLogin" and response code "407"."
 },
 "numOfSystems":0
}
```

Figure 138. Sample response when the authentication request fails

Example

To authenticate with the HTTP proxy server that is between the primary z/OSMF instance and the z/OSMF instance that is running on system *sys057*, submit the following request:

```
POST /zosmf/gateway/logon/proxy HTTP/1.1
Host: zosmf1.yourco.com

{"proxyUserId":"claire","proxyPassword":"abc123","systemName":"sys057"}
```

Figure 139. Sample request to authenticate with an HTTP proxy server

Notification services

The Notification services are provided for both a z/OSMF task and vendor application. These services are used to send a notification in the form of a notification record or email, to a single or multiple recipients. On a successful request, all of the recipients will get the notification in their z/OSMF Notification task as the default destination. A notification can also be sent to the inbox of the user's email account and their mobile application based on preferences.

Table 179. Notification methods

Operation	HTTP method and URI path
"Get all of the notifications received by the current user" on page 259	GET /zosmf/notifications/inbox
"Send a notification from a z/OSMF task, when the content is the message from the bundle file" on page 261	POST /zosmf/notifications/new
"Send a notification and mail from a z/OSMF task or z/OSMF user" on page 264	POST /zosmf/notifications/new
"Send a notification from a third party product" on page 267	POST /zosmf/notifications/new

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a `4nn` code or a `5nn` code. For example, `HTTP/1.1 400 Bad Request` or `HTTP/1.1 500 Internal Server Error`.

In addition, some errors might also include a returned JSON object that contains a message that describes the error. You can use this information to diagnose the error or provide it to IBM Support, if required. For the contents of the error report document, see "Error report document" on page 454.

The following HTTP status codes are valid:

HTTP 200 OK
: Request was processed successfully.

HTTP 400 Bad request
: Request could not be processed because it contains a syntax error or an incorrect parameter.

HTTP 401 Unauthorized
: Request could not be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both, or the client did not authenticate to z/OSMF.

HTTP 404 Not found
: Requested resource does not exist.

HTTP 500 Internal server error
: Server encountered an error. See the response body for a JSON object with information about the error.

Error logging

Errors from the z/OSMF notifications services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide* .

Get all of the notifications received by the current user

You can use this operation to get all of the notifications that were received by the current user. This operation supports only the user to get notification items in the z/OSMF Notifications task. This does not apply to the get mail operation in a user's email account.

HTTP method and URI path

```
GET /zosmf/notifications/inbox
```

Query Parameters

None.

Request

None.

Response Content

On successful completion, the service returns a response body, which contains details about the notifications. Table 180 lists the fields in the response body.

Table 180. Response content from the notifications received by the current user

Field	Type	Description
taskId	String	This is the ID of the task where the notification is initiated.
pluginId	String	This is the ID of the plug-in where the notification is initiated.
appLinkEventId	String	This is the event ID for an application event.
assignees	String	The user IDs and group names of all the recipients. These values are represented as a string, and are separated by a comma.
descriptionParms	String	These are the parameters which substitute the description.
defaultDescription	String	This is the default description of the notification.
descriptionId	String	This is the message ID of the description.
descriptionBundleURL	String	This is the bundle file where the description of the notification can be found.
appLinkParms	String	This is the map of the parameters, which are sent with an event.

Authorization Requirements

Use of the Notification RESTful services API requires the client to be authenticated. For information about client authentication in z/OSMF. See "Authenticating to z/OSMF" on page 1.

You will also need SAF authority, as described in Appendix A in IBM z/OS Management Facility Configuration Guide

HTTP status codes

On successful completion, HTTP status code `200 OK` is returned and the response body is provided, see Table 180 on page 259.

The HTTP status codes and error handling are described in "Error handling" on page 258.

Example HTTP interaction

Request

```
GET /zosmf/notifications/inbox
```

Response

```
{"items":[{"appLinkHandlers":true,
      "taskID":"Workflows",
      "pluginID":"workflow",
      "descriptionBundle":"WorkflowMessages",
      "appLinkEventID":"IBM.ZOSMF.WORKFLOWS.CREATE_WORKFLOW",
      "userRead":false,
      "assignees":"zosmfad",
      "descriptionParms":["testing -service"],
      "defaultDescription":"One or more steps in workflow \"testing -service\" have been assigned to you.",
      "descriptionID":"IZUWF0040I",
      "timestamp":"1429860854757",
      "notificationID":"1429860854756",
      "bundleUrl":"\/zosmf\/workflow\/js\/zosmf",
      "descriptionBundleURL":"\/zosmf\/workflow\/js\/zosmf\/",
      "defaultTaskName":"Workflows",
      "appLinkParms":{"workflow_name":"testing -service"},
      "bundleName":"taskBundle"}],
      "numUnreadNotification":1}
```

Send a notification from a z/OSMF task, when the content is the message from the bundle file

This operation is used to send a notification from a z/OSMF task and the content of the notification is the message from the bundle file.

HTTP method and URI path

```
POST /zosmf/notifications/new
```

Query Parameters

None

Description

The content of the notification contains the message from the bundle file. This operation supports application linking from the notification task to the receiver task. The destination of a notification will depend on user preferences.

Request Content

A notification is sent from a z/OSMF task. The content should include the message ID and the message text, which originate from a bundle file. In this case the value of the post body should be JSON Object-like. See Table 181.

Table 181. Request content for the send notification request

Input	Description	Type	Required or Optional
pluginId	ID of the plug-in where the notification is initiated.	String	Required
taskId	ID of the task where the notification is initiated.	String	Required
assignees	User IDs of all recipients.	String	Required
descriptionBundleURL	Bundle file where the description of the notification can be found.	String	Required
descriptionId	Message ID of the description.	String	Required
defaultDescription	Default description of the notification.	String	Required
descriptionParms	Parameters that substitute the description.	String	Optional
appLinkEventId	Event ID for an application event.	String	Optional
appLinkParms	Map of the parameters, which are sent with an event.	String	Optional

Response Content

On completion, the request returns a JSON object with details about the notification. The response content is shown in Table 182.

Table 182. Response content for the send notification request

Field	Type	Description
apiVersion	String	The version of the Notification Services API.
result	JSONObject	Contains all of the output for each notification destination. It includes 1-3 keys depending on how many destinations the notification is sent to. Each key represents one destination, and its value is a JSONObject which might have messages and return codes.

Authorization Requirements

Use of the Notification RESTful services API requires the client to be authenticated. For information about client authentication in z/OSMF. See "Authenticating to z/OSMF" on page 1.

You will also need SAF authority, as described in Appendix A in IBM z/OS Management Facility Configuration Guide

HTTP status codes

On successful completion, HTTP status code `200 OK` is returned and the response body is provided, Table 182.

The HTTP status codes and error handling are described in "Error handling" on page 258.

Escaping special characters

The format of a notification should be a valid JSONObject. If a special character exists it must be escaped. If " exists, it needs to be escaped as \". If \ exists, it needs to be escaped as \\.

Example HTTP interaction

Request

```
POST /zosmf/notifications/new
{  "pluginId": "workflow",
   "taskId": "Workflows",
   "assignees": "zmfuser, zosmfad, z/OSMF Administrators",
   "descriptionBundleURL": "/zosmf/workflow/js/zosmf/",
   "descriptionId": "IZUWF0039I",
   "defaultDescription": "This is a default description.",
   "applLinkId": "IBM.ZOSMF.WORKFLOWS.CREATE_WORKFLOW",
   "appLinkParams": {"workflow_name": "new workflow"}
data:{
  "event":  {
        "dte":  "15/12/31",
        "tme":  "04:29:57",
        "sys":  "SYS1",
        "cat":  "2",
        "col":  "red",
        "msg":  "event 1 (of event list)",
```

```
          "lng":  "long message",
          "viw":  {"workflows":{"key":"13309779173140.992077"}}
               }
        }
    }
```

Response

```
{"apiVersion":"1.0",
"result":{"mail":{"messages":null,
"rc":"Ok"},
"notification":{"messages":null}}}
}
```

Send a notification and mail from a z/OSMF task or z/OSMF user

This operation is used to send a notification from a z/OSMF task or a z/OSMF user.

HTTP method and URI path

```
POST /zosmf/notifications/new
```

Query Parameters

None

Description

The content of the notification as well as the mail contains the user input data. This operation does not support application linking. A notification with the same subject and content will be sent to all recipients. If the “attachment” parameter is specified, the attachment will only appear in the recipients' mail.

Request Content

The value of the post body should be JSON Object-like. See Table 183.

Table 183. Request content from a notification that requires user input

Input	Description	Type	Required or Optional
assignees	User IDs or groups.	String	Required
subject	Subject of the notification. The allowable length is 1-500.	String	Required
content	Notification body. The allowable length is 0-5000.	String	Optional
attachment	Array of the file paths, up to 5 attachments are allowed.	String	Optional
pluginId	ID of the plug-in where the notification is initiated.	String	Required, only if the notification is from a z/OSMF task.
taskId	ID of the task where the notification is initiated.	String	Required, only if the notification is from a z/OSMF task.
product	Product of Mapping.	String	Required, only if the notification is from a third party product.
eventGroup	Event group of Mapping.	String	Required, only if the notification is from a third party product.
data	Payload content of a notification.	JSON String	Required, only if the notification is from a third party product.
alert	Displayed on APP in the systems that support the alert function.	String	Optional

Response Content

On completion, the request returns a JSON object with details about the notification. The response content is shown in Table 184.

Table 184. Response content from a notification that requires user input

Field	Type	Description
apiVersion	String	The version of the Notification Services API.
result	JSONObject	Contains all of the output for each notification destination. It includes 1-3 keys depending on how many destinations the notification is sent to. Each key represents one destination, and its value is a JSONObject which might have messages and return codes.

Authorization Requirements

Use of the Notification RESTful services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

You will also need SAF authority, as described in Appendix A in IBM z/OS Management Facility Configuration Guide

HTTP status codes

On successful completion, HTTP status code `200 OK` is returned and the response body is provided. See Table 184.

The HTTP status codes and error handling are described in "Error handling" on page 258.

Escaping special characters

The format of a notification should be a valid JSONObject. If a special character exists it must be escaped. If " exists, it needs to be escaped as \". If \ exists, it needs to be escaped as \\.

Example HTTP interaction

Request

```
POST zosmf/notifications/new

{"subject":"Test with unix attachment",
"content":"See if there is an attachment.",
"assignees":"zosmfad",
"attachment":"[\"/var/zosmf/data/logs/IZUG0.log\"]"}
```

Response

```
{"apiVersion":"1.0",
"result":{"mail":{"messages":{"errorData":[{"messageText":"IZUG615E: The
connection to the SMTP host \"smtp.gmail.com\" port \"587\" failed with
```

```
|  error type \"ConnectionTimedout\" .",
|  "messageId":"IZUG615E"}]},
|  "rc":"ConnectionTimedout"},
|  "notification":{"messages":null}}}

|
```

Send a notification from a third party product

This operation is used to send a notification from a third party product. .

HTTP method and URI path

```
POST /zosmf/notifications/new
```

Query Parameters

None

Description

A notification with the same content, which is specified by the element data, is sent to all of the devices that are retrieved by the elements: product and eventGroup. This operation does not support application linking. Mail with the same subject and content will be sent to all recipients. If the “attachment” parameter is specified, the attachment will only appear in the recipients' mail.

Request Content

The value of the post body should be JSON Object-like. See Table 185.

Table 185. Request content from a third party product

Input	Description	Type	Required or Optional
subject	Subject of the notification. The allowable length is 1-500.	String	Required
content	Notification body. The allowable length is 0-5000.	String	Optional
attachment	File path of the attachment.	String	Optional
product	Product of Mapping.	String	Required, only if the notification is from a third party product.
eventGroup	Event group of Mapping.	String	Required
data	Payload content of a notification.	JSON String	Required, only if the notification is from a third party product.
alert	Displayed on APP in the systems that support the alert function.	String	Optional

Response Content

On completion, the request returns a JSON object with details about the notification. The response content is shown in Table 186.

Table 186. Response content from a third party product

Field	Type	Description
apiVersion	String	The version of the Notification Services API.

Table 186. Response content from a third party product (continued)

Field	Type	Description
result	JSONObject	Contains all of the output for each notification destination. It includes 1-3 keys depending on how many destinations the notification is sent to. Each key represents one destination, and its value is a JSONObject which might have messages and return codes.

Authorization Requirements

Use of the Notification RESTful services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

You will also need SAF authority, as described in Appendix A in IBM z/OS Management Facility Configuration Guide

HTTP status codes

On successful completion, HTTP status code `200 OK` is returned and the response body is provided. See Table 186 on page 267.

The HTTP status codes and error handling are described in "Error handling" on page 258.

Escaping special characters

The format of a notification should be a valid JSONObject. If a special character exists it must be escaped. If " exists, it needs to be escaped as \". If \ exists, it needs to be escaped as \\.

Example HTTP interaction

Request

```
POST zosmf/notifications/new

{
    "subject": "to zosmfad 1012",
    "content": "1012-1",
    "eventGroup": "Info",
    "product": "MPF",
    "data": {
            "event": {
                  "dte": "15/12/31",
                  "tme": "04:29:57",
                  "sys": "SYS1",
                  "cat": "2",
                  "col": "red",
                  "msg": "event 1 (of event list)",
                  "lng": "long message",
                  "tok": "",
                  "cmd": ""
                      }
             }
}
```

Response

```
{"apiVersion":"1.0",
    "result":{
          "mobile":{"messages":null,"rc":"Ok"},
          "mail":{"messages":null,"rc":"Ok"},
          "notification":{"messages":null}
              }
}
```

Software management services

The software management REST interface is an application programming interface (API) implemented through industry standard Representational State Transfer (REST) services. This interface allows a client application to interact with the z/OSMF Software Management task.

Table 187 lists the operations that the software management services provide.

Table 187. Operations provided through the software management services.

Operation	HTTP method and URI path
"List the software instances defined to z/OSMF" on page 278	GET /zosmf/swmgmt/swi
"Retrieve the properties of a software instance" on page 281	GET /zosmf/swmgmt/swi/<*system-nickname*>/<*swi-name*>
"List the data sets included in a software instance" on page 286	POST /zosmf/swmgmt/swi/<*system-nickname*>/<*swi-name*>/datasets
"Add a new software instance" on page 292	POST /zosmf/swmgmt/swi
"Export a defined software instance" on page 295	POST /zosmf/swmgmt/swi/<*system-nickname*>/<*swi-name*>/export
"Modify the properties of a software instance" on page 301	PUT /zosmf/swmgmt/swi/<*system-nickname*>/<*swi-name*>
"Load the products, features, and FMIDs for a software instance" on page 305	PUT /zosmf/swmgmt/swi/<*system-nickname*>/<*swi-name*>/products
"Delete a software instance" on page 311	DELETE /zosmf/swmgmt/swi/<*system-nickname*>/<*swi-name*>

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation using the z/OSMF Software Management task. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP response data

The JSON content type (`"Content-Type: application/json"`) is used for response data.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains the following attributes:

```
{
  "error":
  {
    "reason":"reason-code",
    "messages":["message-text"],
    "stack":"stack-trace"
  }
}
```

where:

error JSON object that contains a reason code, a list of one or more message strings to describe the errors detected while processing the request, and for some errors, a stack trace of the exception.

reason-code
Reason code returned for the request. The value is an integer.

message-text
Array that contains the text of each message that was issued.

stack-trace
Stack trace for the exception.

The following HTTP status codes are valid:

HTTP 200 OK
Success.

HTTP 400 Bad request
The request contained incorrect parameters. Table 188 lists the reason codes and messages that can be included in the JSON object returned with HTTP response code 400, and provides a brief description of the error.

Table 188. Reason codes and messages returned for HTTP response code 400

Reason Code	Message	Description
4	The required attribute *attribute-name* is missing. Add the missing attribute and retry the request.	The input JSON object is missing a required attribute.
5	The *object-type* name *object-name* is not valid. Specify a valid name and retry the request.	A software instance or category name specified in the input JSON object is not valid.
6	The global zone CSI data set name *CSI-data-set-name* is not valid. Specify a valid CSI data set name and retry the request.	The CSI data set name specified in the input JSON object is not valid.
7	SMP/E zone name *zone-name* is not valid. Specify a valid target zone name and retry the request.	A target zone name specified in the input JSON object is not valid.
8	The description cannot exceed 256 characters. Specify a valid description and retry the request.	The description specified in the input JSON object exceeds the maximum length.
9	Data set name *data-set-name* is not valid. Specify a valid data set name and retry the request.	A non-SMP/E managed data set name specified in the input JSON object is not valid.
10	Data set volume *volume-serial* is not valid. Specify a valid volume serial and retry the request.	A data set volume specified for a non-SMP/E managed data set in the input JSON object is not valid.
13	Category name *category-name* is a reserved category name. Specify a category name that is not reserved and retry the request.	A category named "NOCATEGORY" was specified in the input JSON object, which is not allowed.

Table 188. Reason codes and messages returned for HTTP response code 400 (continued)

Reason Code	Message	Description
41	The request cannot be performed because the software instance does not have a global zone.	Products, features, and FMIDs cannot be loaded for the subject software instance because the software instance does not have a global zone property.
42	Both of the properties {0} and {1} are missing, but one is required. Add one or both of the missing properties and retry the request.	Every software instance must have at least one of the specified properties. Either add one or both of the missing properties and retry the request.
43	The {0} property requires the {1}property, but the {1} property is missing. Either add the {1} property or remove the {0} property and retry the request.	For every software instance, the first identified property can only be specified if the second identified property is also specified. Either add the second property or remove the first, and retry the request.
45	The package directory {0} is not valid. Specify a valid UNIX directory path and retry the request.	The package directory specified in the input JSON is invalid.
46	The JCL data set name {0} is not valid. Specify a valid data set name and retry the request.	The JCL data set name specified in the input JSON is invalid.
47	JCL card {0} for the JOB statement is not valid. Specify a valid JOB statement and retry the request.	The JOB statement specified in the input JSON is invalid.
48	The UNIX file system data set name{0} is not valid. Specify a valid dataset name and retry the request.	A data set name for a UNIX file system data set in the input JSON is invalid.
49	The UNIX file system mount point {0}for data set {1} is not valid. Specify a valid UNIX path and retry the request.	A mount point for a UNIX file system dataset in the input JSON is invalid.
50	JCL data set {0} is not valid. It must be a PDS or PDS/E with a fixed block record format and a logical record length of 80.	The JCL data set specified in the input JSON exists but has incorrect attributes.

HTTP 401 Unauthorized

The submitter of the request did not authenticate to z/OSMF.

HTTP 403 Forbidden

The server rejected the request. Table 189 lists the reason codes and messages that can be included in the JSON object returned with HTTP response code 403, and provides a brief description of the error.

Table 189. Reason codes and messages returned for HTTP response code 403

Reason Code	Message	Description
1	User *user-ID* is not authorized to use the *task-name* task on system *system-name*.	The specified user is not authorized to use the Software Management task on the indicated system.
11	The software instance cannot be *action-requested* because category *category-name* does not exist and user *user-ID* is not authorized to add categories.	The request to add or update (*action-requested*) the software instance failed because the categories indicated in the message do not exist and the z/OSMF user is not authorized to add categories.

Table 189. Reason codes and messages returned for HTTP response code 403 (continued)

Reason Code	Message	Description
12	The software instance cannot be *action-requested* because user *user-ID* is required to specify one or more categories.	The request to add or update (*action-requested*) the software instance failed because the software instance is not assigned to a category and the z/OSMF user is not authorized to define uncategorized software instances.
14	The action on the software instance cannot be performed because user *user-ID* is not authorized to use category *category-name*.	The z/OSMF user is not authorized to read the specified category.
15	The software instance cannot be *action-requested* because category *category-name* does not exist and user *user-ID* is not authorized to add the category.	The request to add or update (*action-requested*) the software instance failed because the specified category does not exist and the z/OSMF user is not authorized to add the category.
17	The software instance cannot be *action-requested* because global zone *CSI-data-set-name* on system *system-name* does not exist and user *user-ID* is not authorized to add the global zone.	The request to add or update (*action-requested*) the software instance failed because the specified global zone does not exist and the z/OSMF user is not authorized to add the global zone.
18	User *user-ID* is not authorized to perform the requested action on software instance *instance-name* on system *system-name*.	The z/OSMF user is not authorized to perform the requested action on the specified software instance.
35	User *user-ID* is not allowed to obtain the results for this request. Only user *submitter-user-ID*, who submitted the request, is allowed to obtain the results.	When an asynchronous operation is requested, z/OSMF ensures that the user ID that submits the request is authorized to perform the operation. Because authorization checking is not performed when the results are obtained, only the user ID that submitted the request is allowed to obtain its results.

HTTP 404 Not found

The target of the request was not found. Table 190 lists the reason codes and messages that can be included in the JSON object returned with HTTP response code 404, and provides a brief description of the error.

Table 190. Reason codes and messages returned for HTTP response code 404

Reason Code	Message	Description
19	System entry *system-name* is not defined to z/OSMF. Specify the name for an existing z/OSMF system entry and retry the request.	A system entry was not found in the z/OSMF Systems task for the system name specified in the input JSON object.
21	A software instance with the name *instance-name* on system *system-name* does not exist. Specify an existing software instance and retry the request.	The software instance does not exist.
40	The results for the request have expired and are no longer available, or no such request exists.	The service could not find any results for the request identified on the status URL because either the ID specified in the URL is incorrect or the time period for which z/OSMF keeps the results has elapsed, causing the results to expire and be unavailable for the client to obtain.

HTTP 409 Conflict

The request could not be completed because there is a conflict with the current state of the

resource. Table 191 lists the reason codes and messages that can be included in the JSON object returned with HTTP response code 409, and provides a brief description of the error.

Table 191. Reason codes and messages returned for HTTP response code 409

Reason Code	Message	Description
20	The software instance cannot be *action-requested* because a software instance with the name *instance-name* on system *system-name* already exists.	The request to add or update (*action-requested*) the software instance failed because a software instance with the specified name already exists on the specified system. The software instance name and system combination must be unique.
23	The software instance cannot be {added | updated | exported} because it is being deployed or replaced in a deployment that is in progress.	The software instance cannot be updated,deleted or exported because it is being deployed or replaced in a deployment that is in process. You must cancel the in-process deployment before updating,deleting or exporting the software instance.
24	The software instance cannot be {updated | deleted | exported} because it is currently locked by user *user-ID*. It was locked at *date-time*.	A software instance cannot be updated,deleted, or exported while it is currently locked and being updated by another user. The date and time when it was locked is given. The software instance must be unlocked before attempting to update,delete or export it.
52	The software instance cannot be exported because user {0} is exporting the same software instance and has stolen the lock. Try the request again later.	A software instance cannot be exported while it is being exported by another user. The current export operation must complete before attempting to export the software instance again.

HTTP 500 Internal server error

The server encountered an error that prevented it from completing the request. Table 192 lists the reason codes and messages that can be included in the JSON object returned with HTTP response code 500, and provides a brief description of the error.

Table 192. Reason codes and messages returned for HTTP response code 500

Reason Code	Message	Description
2	The request could not be completed because an error occurred. Error: *error-details*	An unexpected error occurred while processing the request. The details of the error (*error-details*) are provided in the message.
22	The maximum number of products, features, and FMIDs that can be returned for a software instance was exceeded.	More than 10,000 products, features, and FMIDs exist for the specified software instance; however, only 10,000 or fewer can be returned.
25	The request failed because system entry *system-name* is not defined to z/OSMF.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the specified system does not exist in the z/OSMF Systems task. Define the system to the primary z/OSMF instance, and retry the request.

Table 192. Reason codes and messages returned for HTTP response code 500 (continued)

Reason Code	Message	Description
26	A secure connection to z/OSMF host system *system-name* could not be established because the certificate for the specified system is not trusted or has changed.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the certificate for the secondary z/OSMF instance is not trusted or has changed.
27	A secure connection to z/OSMF host system *system-name* could not be established because an error occurred. Error: *error-details*	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because an error occurred. Additional information about the error is provided in the message text.
28	The attempt to connect to z/OSMF host system *system-name* failed, perhaps due to a momentary loss of connectivity. If this error persists, however, a more serious connectivity problem might exist in your network.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because a connectivity error occurred. Try the request again later.
29	The request was not completed because the connection to z/OSMF host system *system-name* timed out.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the connection to that system timed out.
30	The attempt to authenticate with z/OSMF host system *system-name* failed because the security credentials provided are not valid.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the user ID and password or passphrase specified for the secondary z/OSMF instance in the input JSON object are not valid. Specify the correct user ID and password or passphrase, and retry the request.
31	The attempt to connect to z/OSMF host system *system-name* failed because the security credentials provided for proxy server *proxy-name* are not valid.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the user ID and password specified for the HTTP proxy server in the input JSON object are not valid. Specify the correct user ID and password, and retry the request.
32	The attempt to authenticate with z/OSMF host system *system-name* failed because security credentials are required, but none were provided.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the user ID and password for the secondary z/OSMF instance were not specified in the input JSON object. Specify the correct user ID and password, and retry the request.

Table 192. Reason codes and messages returned for HTTP response code 500 (continued)

Reason Code	Message	Description
33	The attempt to connect to z/OSMF host system *system-name* failed because security credentials for proxy server *proxy-name* are required, but none were provided.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because the user ID and password for the HTTP proxy server were not specified in the input JSON object. Specify the correct user ID and password, and retry the request.
34	A connection to z/OSMF host system *system-name* could not be established because an error occurred. See the z/OSMF logs for additional details on the cause of the error.	The primary z/OSMF instance cannot connect to the secondary z/OSMF instance that resides in the same sysplex as the software instance because an error occurred. Examine the z/OSMF logs to obtain more information about the cause of this problem.
36	*SMP/E-messages*	The request could not be completed because one or more errors were detected by SMP/E. The messages returned from SMP/E are provided. Resolve any errors, and retry the request.
37	The product, feature, and FMID information was not loaded because the software instance was recently modified. Try the request again.	The software instance was modified by another user while the retrieve product, feature, and FMID information request for the software instance was in progress. The results of the retrieve operation cannot be trusted and have been discarded. Try the request again after the modify operation is completed.
38	z/OSMF on system *system-name* is not at the required service level to communicate with a higher level of z/OSMF on other systems. z/OSMF FMID *fmid* on system *system-name* requires APAR *apar-number*.	The requested action requires that z/OSMF communicate with instances of z/OSMF on other systems. However, the z/OSMF instance on the indicated system is not at a service level that supports communicating with higher levels of z/OSMF on other systems. The FMID for the z/OSMF plug-in that requires service is indicated in the message. To allow communication between the z/OSMF instances, ensure that the z/OSMF instances are at compatible service levels. To do so, install the PTF for the indicated APAR onto the indicated system.

Table 192. Reason codes and messages returned for HTTP response code 500 (continued)

Reason Code	Message	Description
39	z/OSMF FMID *fmid* on system *system-name* is not at the required software level to communicate with a higher level of z/OSMF on other systems.	The requested action requires that z/OSMF communicate with instances of z/OSMF on other systems. However, the z/OSMF instance on the indicated system is not at a release level that supports communicating with higher levels of z/OSMF on other systems. The FMID for the z/OSMF plug-in that requires a software update is indicated in the message. To allow communication between the z/OSMF instances, ensure that the z/OSMF instances are at compatible software levels. To do so, upgrade the release of z/OSMF on the indicated system.
51	An I/O exception occurred: {0}	The request could not be completed because an I/O error exception has occurred. Details about the error are provided.

HTTP 503 Service unavailable

The server is currently unavailable to process the request. Table 193 lists the reason codes and messages that can be included in the JSON object returned with HTTP response code 503, and provides a brief description of the error.

Table 193. Reason codes and messages returned for HTTP response code 503

Reason Code	Message	Description
3	z/OSMF is unable to process your request because z/OSMF is busy processing similar requests for other users. Try the request again later.	Multiple users are attempting to access the same database tables and are waiting for each other's transaction to complete, causing a database deadlock condition to occur.

Error logging

Errors from the software management services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

List the software instances defined to z/OSMF

You can use this operation to obtain a list of the software instances that are defined to a z/OSMF instance.

HTTP method and URI path

```
GET /zosmf/swmgmt/swi
```

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.

Standard headers

Use the following standard HTTP header with this request:

Accept-Language
: Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, then English is used.

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation using the z/OSMF Software Management task. That is, to obtain a list of the software instances that are defined to a z/OSMF instance, the user ID initiating the request must have READ access to the z/OSMF Software Management task. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 270.

If the request is successful, the response also includes the following JSON object:

```
{
  "swilist":[
  {
    "name":"swi-name",
```

```
    "system":"system-nickname",
    "description":"swi-description",
    "globalzone":"global-zone",
    "targetzones":["target-zones"],
    "categories":"["categories"],
    "productinforetrieved":"last-retrieved",
    "lastmodified":"last-modified",
    "modifiedby":"modified-user-ID",
    "created":"date-created",
    "createdby":"created-user-ID",
    "locked":"date-locked",
    "lockedby":"locked-user-ID",
    "swiurl":"swi-URL"
  }
]}
```

where:

swilist
Array that contains each software instance that is defined to z/OSMF.

swi-name
Name of the software instance.

system-nickname
Nickname of the z/OSMF host system that has access to the volumes and data sets where the software instance resides. To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

swi-description
Description of the software instance.

global-zone
CSI data set that contains the global zone used to manage the software. If the software instance has no global zone, then this property will be null.

target-zones
Comma-separated list of the target zones included in the software instance. If the software instance has no global zone, then this property will be null.

categories
Comma-separated list of the categories to which the software instance is assigned.

last-retrieved
Date and time the product, feature, and FMID information was last retrieved for the software instance. This attribute is blank if this information has not been retrieved.

last-modified
Date and time in ISO 8601 format that the software instance was last modified.

modified-user-ID
User ID of the user who last modified the software instance.

date-created
Date and time in ISO 8601 format that the software instance was created.

created-user-ID
User ID of the user who created the software instance.

date-locked
Date and time in ISO 8601 format that the software instance was locked. This attribute is null if the software instance is not currently locked.

locked-user-ID
User ID of the user who locked the software instance. This attribute is null if the software instance is not currently locked.

swi-URL
URL that allows you to access the software instance. For example, a client application can use the URL to read a software instance. For more details, see "Retrieve the properties of a software instance" on page 281.

Example

In the following example, the GET method is used to retrieve a list of the software instances that are defined to the z/OSMF instance that has a host name of *zosmf1.yourco.com*.

```
GET /zosmf/swmgmt/swi HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 140. Sample request to retrieve a list of software instances

A sample response is shown in Figure 141.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Content-Type: application/json
Content-Language: en
Connection: close

{"swilist":[
{"name":"DB2V9", "system":"PEV174", "description":null,
"globalzone":"DB2.GLOBAL.CSI", "targetzones":["DB2TGT"], "categories":null,
"productinforetrieved":"2014-08-20T19:23:25+00:00", "lastmodified":"2014-08-
20T19:23:25+00:00", "modifiedby":"FRED", "created":"2014-08-20T19:23:25+00:00",
"createdby":"BARNEY", "locked":null, "lockedby":null,
"swiurl":"https:\/\/zosmf1.yourco.com\/zosmf\/swmgt\/swi\/PEV174\/DB2V9"}
{"name":"zOSV2R1", "system":"PEV174", "description":null,
"globalzone":"ZOS.GLOBAL.CSI", "targetzones":["MVST100","MVST110"],
"categories":null, "productinforetrieved":"2014-08-20T19:23:25+00:00",
"lastmodified":"2014-08-20T19:23:25+00:00", "modifiedby":"WILMA",
"created":"2014-08-20T19:23:25+00:00", "createdby":"BETTY", "locked":null,
"lockedby":null,
"swiurl":"https:\/\/zosmf1.yourco.com\/zosmf\/swmgt\/swi\/PEV174\/zOSV2R1"}
]}
```

Figure 141. Sample response from a request to retrieve a list of software instances

Retrieve the properties of a software instance

You can use this operation to retrieve the properties of a software instance. The properties include, but are not limited to, the global zone and target zones associated with the software instance and a list of the products, features, FMIDs, and non-SMP/E managed data sets that are included in the software instance.

HTTP method and URI path

```
GET /zosmf/swmgmt/swi/<system-nickname>/<swi-name>
```

---

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.
- ***<system-nickname>/<swi-name>*** further qualifies the request and indicates the specific software instance to be retrieved. A software instance is uniquely identified by its name (*swi-name*) and the nickname (*system-nickname*) of the z/OSMF host system that has access to the volumes and data sets where the software instance resides.

  To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

Standard headers

Use the following standard HTTP header with this request:

Accept-Language
: Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, then English is used.

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation using the z/OSMF Software Management task. That is, to retrieve the properties of a software instance, the user ID initiating the request must have READ access to both the z/OSMF Software Management task and the software instance. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 270.

If the request is successful, the response also includes the following JSON object:

```
{
  "name":"swi-name",
  "system":"system-nickname",
  "description":"swi-description",
  "globalzone":"global-zone",
  "targetzones":["target-zones"],
  "categories":"["categories"],
  "productinforetrieved":"last-retrieved",
  "lastmodified":"last-modified",
  "modifiedby":"modified-user-ID",
  "created":"date-created",
  "createdby":"created-user-ID",
  "locked":"date-locked",
  "lockedby":"locked-user-ID",
  "datasets":[
  {
    "dsname":"data-set-name",
    "volume":"volume-serial"
  }],
  "products":[
  {
    "prodname":"product-name",
    "prodid":"product-ID",
    "release":"product-level",
    "vendor":"vendor-name",
    "generalavailability":"general-availability",
    "endofservice":"end-of-service",
    "url":"product-URL",
    "productinfofileversion":"file-version",
    "features":[
    {
      "feature":"feature-name",
      "fmids":[
      {
        "fmid":"fmid-name",
        "description":"fmid-description",
        "targetzones":["fmid-target-zones"]
      }]
    }]
  }]
}
```

where:

swi-name
: Name of the software instance.

system-nickname
: Nickname of the z/OSMF host system that has access to the volumes and data sets where the software instance resides. To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

swi-description
: Description of the software instance.

global-zone
CSI data set that contains the global zone used to manage the software. If the software instance has no global zone, then this property will be null.

target-zones
Comma-separated list of the target zones included in the software instance. If the software instance has no global zone, then this property will be null.

categories
Comma-separated list of the categories to which the software instance is assigned.

last-retrieved
Date and time the product, feature, and FMID information was last retrieved for the software instance. This attribute is blank if this information has not been retrieved.

last-modified
Date and time in ISO 8601 format that the software instance was last modified.

modified-user-ID
User ID of the user who last modified the software instance.

date-created
Date and time in ISO 8601 format that the software instance was created.

created-user-ID
User ID of the user who created the software instance.

date-locked
Date and time in ISO 8601 format that the software instance was locked. This attribute is null if the software instance is not currently locked.

locked-user-ID
User ID of the user who locked the software instance. This attribute is null if the software instance is not currently locked.

datasets
Array that contains each non-SMP/E managed data set that is included in the software instance.

data-set-name
Name of the non-SMP/E managed data set.

volume-serial
Volume on which the non-SMP/E managed data set resides.

products
Array that contains each product that is included in the software instance.

product-name
Name of the product. If any FMIDs are not related to a product and feature, those FMIDs are listed under a product named *No Product*.

product-ID
Identifier of the product.

product-level
Version, release, and modification level of the product. The value has the format *VV.RR.MM*, where *VV* is the two-digit version, *RR* is the two-digit release, and *MM* is the two-digit modification level.

vendor-name
Name of the vendor that provides the product.

general-availability
Date this level of the product is available to all users.

end-of-service
Last date on which the vendor will deliver standard support services for this level of the product. This date is the general end of service date. It does not account for lifecycle extensions.

product-URL
URL that links to additional information about the product. This information can include, for example, product life cycle dates, product highlights, planning information, and technical descriptions.

file-version
Version of the most recent product information file that was retrieved that contains the corresponding product. The version represents the date that file was created or last updated.

features
Array that contains each feature that is included in the product.

feature-name
Name of the feature. If any FMIDs are not related to a product and feature, those FMIDs are listed under a feature named *No Feature*.

fmids Array that contains each FMID that is included in the feature.

fmid-name
Name of the FMID.

fmid-description
Description of the FMID.

fmid-target-zones
Name of the target zones where the FMID is installed.

Example

In the following example, the GET method is used to retrieve the properties of software instance DB2V9 on system PEV174.

```
GET /zosmf/swmgmt/swi/PEV174/DB2V9 HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 142. Sample request to retrieve the properties of a software instance

A sample response is shown in Figure 143 on page 285.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Content-Type: application/json
Content-Language: en
Connection: close

{
"name":"DB2V9", "system":"PEV174", "description":null,
"globalzone":"DB2.GLOBAL.CSI", "targetzones":["DB2TGT"], "categories":null,
"productinforetrieved":"2014-08-20T19:23:25+00:00", "lastmodified":"2014-08-
20T19:23:25+00:00", "modifiedby":"FRED", "created":"2014-08-20T19:23:25+00:00",
"createdby":"BARNEY", "locked":null, "lockedby":null, "datasets":
[{"dsname":"USER.DB2V9.PROCLIB", "volume": "LV1234"},
{"dsname":"USER.DB2V9.SAMPLES", "volume":"LV1234"}], "products":
[{"prodname":"DB2 for z/OS", "prodid":"5635-DB2", "release":"09.01.00",
"vendor":"IBM", "generalavailability":"20006-06-09T19:23:25+00:00",
"endofservice":"2014-06-27T19:23:25+00:00", "url":null,
"productinfofileversion":"2014-01-01", "features":[{"feature":"DB2 Base",
"fmids":[{"fmid":"HDB9910", "description":"DB2 BASE/TSO", "targetzones":
["DB2V9T"]}]}]}]
}
```

Figure 143. Sample response from a request to retrieve the properties of a software instance

List the data sets included in a software instance

You can use this operation to obtain a list of the data sets that compose a software instance.

HTTP method and URI path

```
POST /zosmf/swmgmt/swi/<system-nickname>/<swi-name>/datasets
```

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.
- ***<system-nickname>/<swi-name>*** further qualifies the request and indicates the specific software instance to be retrieved. A software instance is uniquely identified by its name (*swi-name*) and the nickname (*system-nickname*) of the z/OSMF host system that has access to the volumes and data sets where the software instance resides.

  To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.
- **datasets** indicates that the data sets included in the software instance are to be obtained.

If the subject software instance has a global zone, z/OSMF analyzes the global, target, and distribution zones included in the software instance to identify the SMP/E managed data sets and the SMP/E managed UNIX data sets that contain the installed software described in those zones. z/OSMF returns a JSON object that lists the properties of each data set it identified, along with the properties of each non-SMP/E managed data set included in the software instance.

If the subject software instance has no global zone, this method will obtain detailed properties foreach of the non-SMP/E managed data sets that are defined for the software instance, and returnthose data set properties as the results of this method.

Standard headers

Use the following standard HTTP header with this request:

Content-Type
: Identifies the type of input content provided by the caller. Use the JSON content type ("Content-Type: application/json") if a JSON document is included as input with this request.

Accept-Language
: Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, English is used.

Custom headers

None.

Request content

If the software instance does not reside in the same sysplex as the primary z/OSMF instance, you might be required to authenticate with the secondary z/OSMF instance that is running in the sysplex where the software instance resides. In addition, if the primary z/OSMF instance must navigate an HTTP proxy server to connect with the secondary z/OSMF instance, you might also be required to authenticate with that HTTP proxy server. To do so, include the following JSON object in your request:

```
{
  "zosmfuid":"zosmf-user-ID",
  "zosmfpw":"zosmf-password",
  "proxyuid":"proxy-user-ID",
  "proxypw":"proxy-password"
}
```

Figure 144. Request content to authenticate with a secondary z/OSMF instance and an HTTP proxy server

where:

zosmf-user-ID
User ID for authenticating with the secondary z/OSMF instance.

zosmf-password
Password for authenticating with the secondary z/OSMF instance.

proxy-user-ID
User ID for authenticating with the HTTP proxy server.

proxy-password
Password for authenticating with the HTTP proxy server.

Include the JSON object in the request only if you are required to authenticate with a secondary z/OSMF instance or an HTTP proxy server. Otherwise, omit the JSON object.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation using the z/OSMF Software Management task. That is, the user ID must have READ access to both the Software Management task and the SAF resources required to process the request. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request was accepted. If the request was accepted, the service returns status code `202 Accepted` and a JSON object that contains a URL (`{"statusurl":"url"}`). To monitor the status of the list data sets request and to retrieve the results, perform GET requests to the supplied URL. Only the user ID that initiates the list data sets request is authorized to check the status and retrieve the results. One of the following responses is returned from the get status request:

- If the list data sets request is still in progress, an HTTP response code of `200 OK` is returned, along with the following JSON object: `{"status":"status", "percentcomplete":"percent-complete"}`.
- If the list data sets request is complete, an HTTP response code of `200 OK` is returned, along with the following JSON object:

```
{
  "status":"status",
  "percentcomplete":"percent-complete",
  "swidatasets":{
    "smpemanageddatasets":[{
      "dsname":"data-set-name",
      "volumes":["volume-serial"],
```

```
      "dstype":"data-set-type",
      "recfm":"record-format",
      "lrecl":"logical-record-length",
      "blksize":"block-size",
      "tracks":"allocated-tracks",
      "used":"used-tracks-percent",
      "extents":"allocated-extents",
      "zones":["zones"],
      "zonedddefs":[{
        "zone":"zone-name", "dddefs":["dddef-name"]}],
      "msgs":["message-text"]
    }],
    "smpemanagedunixdatasets":[{
      "dsname":"data-set-name",
      "unixdirs":["UNIX-directory"],
      "volumes":["volume-serial"],
      "dstype":"data-set-type",
      "recfm":"record-format",
      "lrecl":"logical-record-length",
      "blksize":"block-size",
      "tracks":"allocated-tracks",
      "used":"used-tracks-percent",
      "extents":"allocated-extents",
      "zones":["zones"],
      "zonedddefs":[{
        "zone":"zone-name", "dddefs":["dddef-name"]}],
      "msgs":["message-text"]
    }],
    "nonsmpemanageddatasets":[{
      "dsname":"data-set-name",
      "volumes":["volume-serial"],
      "dstype":"data-set-type",
      "recfm":"record-format",
      "lrecl":"logical-record-length",
      "blksize":"block-size",
      "tracks":"allocated-tracks",
      "used":"used-tracks-percent",
      "extents":"allocated-extents",
      "msgs":["message-text"]
    }]
  }
}
```

where:

status Status of the list data sets request. The status is either *running* or *complete*.

percent-complete
Percentage of the processing that is complete for the list data sets request, expressed as a whole number from 0 to 100.

swidatasets
Lists all the data sets included in the software instance. A software instance can contain:

- **smpemanageddatasets**. Array of the SMP/E managed data sets included in the software instance. If the software instance has no global zone, then this property will be null.
- **smpemanagedunixdatasets**. Array of the SMP/E managed UNIX file system data sets included in the software instance. If the software instance has no global zone or no SMP/E managed UNIX data sets, then this property will be null.
- **nonsmpemanageddatasets**. Array of the non-SMP/E managed data sets included in the software instance.

data-set-name
Name of the data set.

UNIX-directory
Array of the UNIX directories contained in the data set. If z/OSMF could not identify the UNIX file system data set for any UNIX directories, the data set name for those directories is set to *No Data Set Found*. Typically, this occurs when the UNIX file system data set is not mounted.

volume-serial
Array of the volume serials for the volumes on which the data set resides. If the data set is migrated, a value of *MIGRAT* is returned.

data-set-type
Type of data set. The data set can be one of the following types:
- **HFS**. Hierarchical file system.
- **PDS**. Partitioned data set.
- **PDSE**. Partitioned data set extended.
- **Sequential**. Sequential data set.
- **VSAM**. VSAM data set.
- **ZFS**. zSeries file system.

record-format
Record format specified when the data set was allocated. The record format can be any valid combination of the following codes:
- **A**. ASA printer control characters.
- **B**. Blocked records.
- **F**. Fixed-length records.
- **M**. Machine code printer control characters.
- **S**. Standard (for F) or spanned (for V); used only with sequential data sets.
- **T**. Track-overflow feature.
- **U**. Undefined format records.
- **V**. Variable-length records.

logical-record-length
Logical record length, in bytes, specified when the data set was allocated.

block-size
Block size, in bytes, specified when the data set was allocated.

allocated-tracks
Number of tracks allocated to the data set.

used-tracks-percent
Percentage of allocated tracks used, expressed in whole numbers, not rounded. If any track is used, the minimum percentage is 1. If the data set is a PDSE, the percentage refers to the percentage of allocated pages used.

allocated-extents
Number of extents allocated to the data set.

zones Array of the SMP/E zones that contain a DDDEF entry for the data set. For an SMPCSI data set, it is an array of the SMP/E zones contained in the data set.

zonedeffs
Array of SMP/E zones and DDDEF entries for the data set.

zone-name
Name of an SMP/E zone that contains one or more DDDEF entries for the data set.

dddef-name
Array of DDDEF entry names that identify the data set.

message-text
Array of messages returned for the data set.

- If the list data sets request is complete but the results are no longer available, an HTTP response code of `404 Not found` is returned. z/OSMF makes the results available for a client application for a finite period of time. When that time elapses, the results are no longer available; in which case, the client must reissue the request.

If the list data sets request cannot be processed, a status code of 4*nn* or 5*nn* is returned, indicating that an error has occurred. For more details, see "Error handling" on page 270.

Example

In the following example, the POST method is used to retrieve a list of the data sets included in software instance *DB2V9* on system *SYS123*.

```
POST /zosmf/swmgmt/swi/SYS123/DB2V9/datasets HTTP/1.1
Host: sys123.yourco.com
```

Figure 145. Sample request to list the data sets included in a software instance

Figure 146 provides a sample response, indicating that the list data sets request has been accepted and supplying the URL to use for monitoring the status of that request.

```
HTTP/1.1 202 Accepted
Date: Tues, 21 November 2014 18:53:04 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"statusurl":"https:\/\/sys123.yourco.com\/zosmf\/swmgt\/statusmonitor\/dslist
\/4837290198343"}
```

Figure 146. Sample response for a list data sets request

To check the status of the list data sets request, submit the following request:

```
GET /zosmf/swmgmt/statusmonitor/dslist/4837290198343 HTTP/1.1
Host: sys123.yourco.com
```

Figure 147. Sample request to obtain the status of a list data sets request

Figure 148 provides a sample response for when the list data sets request is in progress.

```
HTTP/1.1 200 OK
Date: Tues, 21 November 2014 18:53:19 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"status":"running", "percentcomplete":"65"}
```

Figure 148. Sample get status response when the list data sets request is in progress

Figure 149 on page 291 provides a sample response for when the list data sets request is complete.

```
HTTP/1.1 200 OK
Date: Tues, 21 November 2014 18:53:27 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"status":"complete", "percentcomplete":"100", "swidatasets":{
"smpemanageddatasets":[
{"dsname":"USERID.SMPE.CSI", "volumes":["SL730C"], "dstype":"VSAM",
"recfm":null, "lrecl":null, "blksize":null, "tracks":"1509", "used":null,
"extents":null, "zones":["GLOBAL", "TGT0", "DLB0"], "zonedddefs":null,
"msgs":null},
{"dsname":"USERID.SMPE.MACLIB", "volumes":["SL7334"], "dstype":"PDS",
"recfm":"FB", "lrecl":"80", "blksize":"27920", "tracks":"4", "used":"100",
"extents":"1", "zones":["TGT0"], "zonedddefs":[{"zone":"TGT0", "dddefs":
["MACLIB", "SMPMTS"]}], "msgs":null},
{"dsname":"USERID.SMPE.MIGLIB", "volumes":["SL8B2D"], "dstype":"PDS",
"recfm":"U", "lrecl":null, "blksize":"32760", "tracks":"147", "used":"100",
"extents":"1", "zones":["TGT0"], "zonedddefs":[{"zone":"TGT0", "dddefs":
["MIGLIB"]}], "msgs":null},
{"dsname":"USERID.SMPE.SGIMCLS0", "volumes":["SL7307"], "dstype":"PDS",
"recfm":"VB", "lrecl":"255", "blksize":"32760", "tracks":"15", "used":"6",
"extents":"1", "zones":["TGT0"], "zonedddefs":[{"zone":"TGT0", "dddefs":
["SGIMCLS0"]}], "msgs":null},
{"dsname":"USERID.SMPE.SGIMMJPN", "volumes":["MIGRAT"], "dstype":null,
"recfm":null, "lrecl":null, "blksize":null, "tracks":null, "used":null,
"extents":null, "zones":null, "zonedddefs":null, "msgs": ["GIM70531E Data set
USERID.SMPE.SGIMMJPN is migrated and is being recalled."]},
...],
"smpemanagedunixdatasets":[{"dsname":"USERID.SMPE.ZFS", "unixdirs":
["/u/userid/smpe/usr/lpp/smp/IBM/"], "volumes":["ZF3804"], "dstype":"ZFS",
"recfm":null, "lrecl":null, "blksize":null, "tracks":"22650", "used":null,
"extents":null, "zones":["TGT0"], "zonedddefs":[{"zone":"TGT0", "dddefs":
["SGIMBIN"]}], "msgs":null],
"nonsmpemanageddatasets":null
}}
```

Figure 149. Sample get status response when the list data sets request is complete

Add a new software instance

You can use this operation to add a software instance to z/OSMF.

HTTP method and URI path

```
POST /zosmf/swmgmt/swi
```

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.

Standard headers

Use the following standard HTTP header with this request:

Content-Type
Identifies the type of input content provided by the caller. The JSON content type ("Content-Type: application/json") is used for the JSON document included as input with this request.

Accept-Language
Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, English is used.

Custom headers

None.

Request content

Your request must include a JSON object that describes the software instance to be added, for example:

```
{
  "name":"swi-name",
  "system":"system-nickname",
  "description":"swi-description",
  "globalzone":"global-zone",
  "targetzones":["target-zones"],
  "categories":"["categories"],
  "datasets":[
  {
    "dsname":"data-set-name",
    "volume":"volume-serial"
  }]

}
```

Figure 150. Adding a software instance: request content

where:

swi-name
Name of the software instance. The name can contain up to 30 non-blank characters, including alphanumeric characters (A-Z, a-z, and 0-9), mathematical symbols (< > - = | \), punctuation marks (? ! : ' " /), and special characters ($ _ # @ ^). The name is required and must be unique on the system.

system-nickname
Nickname of the system that has access to the volumes and data sets where the software instance resides. Use the nickname that is specified for the system definition in the z/OSMF Systems task. The nickname is required.

To manage the systems defined to z/OSMF, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

swi-description
Description of the software instance. The description is optional and can contain a maximum of 256 characters.

global-zone
CSI data set that contains the global zone used to manage the software. If specified, must comply with z/OS data set naming conventions, and must end with *.CSI*.

target-zones
Comma-separated list of the target zones to be included in the software instance. A list of 0 or more zone names, each target zone name must be 1-7 characters long; the valid characters are uppercase alphabetic characters (A-Z), numeric characters (0-9), and special characters (@ # $). The first character must be an alphabetic character.

categories
Comma-separated list of the categories to which the software instance is assigned. Each category name can contain up to 30 non-blank characters, including alphanumeric characters (A-Z, a-z, and 0-9), mathematical symbols (< > - = | \), punctuation marks (? ! : ' " /), and special characters ($ _ # @ ^). Assigning the software instance to a category is optional.

datasets
Array that contains each non-SMP/E managed data set to be added to the software instance. Adding non-SMP/E managed data sets to the software instance is optional.

data-set-name
Name of the non-SMP/E managed data set. You cannot specify data set members or a subset of a data set. A data set name is required if you are adding non-SMP/E managed data sets to the software instance. The data set name must comply with z/OS data set naming conventions

volume-serial
Volume on which the non-SMP/E managed data set resides. The volume serial is required if the data set is not cataloged, and the volume must be accessible by the system where the software instance resides. The volume serial must be 6 characters long; the valid characters are uppercase alphabetic characters (A-Z) and numeric characters (0-9).

When adding a software instance, the software management REST interface validates the information that is supplied in the JSON object before completing the request. All input values must be valid and syntactically correct. If any errors are discovered, the add request fails and appropriate messages are returned.

The name and system, properties are required for every software instance. If either of these are not specified or are specified as null, then the request finishes with HTTP response code 400 (Badrequest) and reason code 4 (The required property {name|system} is missing. Add the missing property and retry the request.).

If the globalzone property is specified, then the targetzones property is required. Therefore, if globalzone is specified and targetzones is not, then the request finishes with HTTP response code400 (Bad request) and reason code 4 (The required property {targetzones} is missing. Add the missing property and retry the request.).

If the globalzone property is not specified, then the datasets property is required. That is, the software instance must contain either SMP/E managed software, or at least one non-SMP/Emanaged data set.

Therefore, if both globalzone and datasets are not specified or are specified as null, then the request finishes with HTTP response code 400 (Bad request) and reason code 42 (Both of the properties {globalzone} and {datasets} are missing, but one is required. Add one or both of the missing properties and retry the request.).

The targetzone property can only be specified if the globalzone property is also specified. Therefore, if targetzone is specified but globalzone is not, then the request finishes with HTTP response code 400 (Bad request) and reason code 43 (The {targetzone} property requires the{globalzone} property, but the {globalzone} property is missing. Either add the {globalzone}property or remove the {targetzone} property and retry the request.)

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation using the z/OSMF Software Management task. That is, the user ID must have READ access to the Software Management task, and CONTROL access to the SAF resources corresponding to the software instance to be added. If the specified categories are implicitly added during this software instance add operation, the user ID must also have CONTROL access to the SAF resources corresponding to the specified categories. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 270.

Example

In the following example, the POST method is used to add a software instance to the z/OSMF instance that has a host name of *pev174.yourco.com*.

```
POST /zosmf/swmgmt/swi HTTP/1.1
Host: pev174.yourco.com

Content-Type: application/json
Accept-Language: en
{
  "name":"DB2V9",
  "system":"PEV174",
  "description":"DB2",
  "globalzone":"DB2.GLOBAL.CSI",
  "targetzones":["DB2TGT"],
  "datasets":[
  {"dsname":"USER.DB2V9.PROCLIB","volume": "LV1234"},
  {"dsname":"USER.DB2V9.SAMPLES","volume":"LV1234"}]
}
```

Figure 151. Sample request to add a software instance

Export a defined software instance

A portable software instance is a set of portable files that represent the content of a z/OSMF software instance. An Export action on a software instance is used to create a portable software instance. You can use the POST method to perform an Export action on a software instance that is defined to z/OSMF, which generates a portable software instance descriptor file and JCL that when executed creates the archive files for a portable software instance, and store those files in a UNIX directory on the system where the software instance being exported resides.

HTTP method and URI path

```
POST /zosmf/swmgmt/swi/<system-nickname>/<swi-name>/export
```

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.
- ***<system-nickname>/<swi-name>*** further qualifies the request and indicates the specific software instance to be exported. A software instance is uniquely identified by its name (*swi-name*) and the nickname (*system-nickname*) of the z/OSMF host system that has access to the volumes and data sets where the software instance resides.

  To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.
- **/export** indicates JCL to perform an export action for the software instance is to be generated.

Standard headers

Use the following standard HTTP headers with this request:

Accept-Language
: Identifies the preferred language for messages that can be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, then English is used.

Content-Type
: Identifies the type of input content that is provided by the caller. The JSON content type("Content-Type: application/json") is used for the JSON document, if any, included as input with this request.

Custom headers

None.

Request content

The request must include a JSON document that identifies properties that are required to perform the operation. For example:

```
{
 "packagedir":"UNIX-directory",
 "jcldataset":"data-set-name",
 "includedlibs":"yes | no",
 "jobstatement":["jclrecord"],
 "unixdatasets":[{
   "dsname":"data-set-name",
   "mountpoint":"UNIX-path"
```

```
    }],
   "zosmfuid":"user-id",
   "zosmfpw":"password",
   "proxyuid":"user-id",
   "proxypw":"password"
 }
```

where:

packagedir
UNIX directory to contain the files for the portable software instance.

Must be a UNIX directory with valid UNIX directory name syntax:

- Must be an absolute pathname (must start with slash).
- Must end with a slash.
- Can be up to 1023 characters long.

jcldataset
Partitioned data set to contain the portable software instance descriptor file and the generated JCL.

The data set name must comply with z/OS data set naming conventions.

includedlibs
If the software instance contains SMP/E managed software, this property indicates whether the distribution libraries and DLIB zones are to be included in the portable software instance.

This is an optional property. Can be null, “yes” or “no”, but the default value is “yes”.

jobstatement
List of JCL cards for the JOB statement to be used in the generated JCL for the export operation.

This is an optional property. Can be null, or a list of JCL cards, each up to 72 characters long. Columns 1 and 2of each record must be “//” or “/*” and the job name must be 1 to 8 characters. If no JOB statement is provided, the default is exactly://IZUD01EX JOB (ACCOUNT),'NAME'.

unixdatasets
List of UNIX file system data sets in the software instance that are currently not mounted, and therefore, cannot be identified by z/OSMF by referencing the UNIX directories that are defined in the SMP/E target zone DDDEF entries.

This is an optional property, required only if the software instance describes SMP/E managed software, and if any of the UNIX file system data sets containing that software are currently not mounted.

dsname
Name of a UNIX file system data set.

The data set name must comply with z/OS data set naming conventions.

mountpoint
Mount point for the UNIX file system data set, if the data set were mounted.

Must have valid UNIX file name syntax:

- Must be an absolute pathname (must start with slash).
- Must not end with a slash, unless a slash is the only character. That is, “/” (root) is valid, but “/mountpoint/” is not.
- Can be up to 1023 characters long.

zosmfuid
Userid for authenticating with a remote z/OSMF instance.

This is an optional property.

zosmfpw
Password for authenticating with a remote z/OSMF instance.

This is an optional property

proxyuid
Userid for authenticating with an HTTP proxy.

This is an optional property.

proxypw
Password for authenticating with an HTTP proxy.

This is an optional property.

The request content is required, but some properties are optional. For example, if the software instance does not reside in the same sysplex as the primary z/OSMF instance, you might be required to authenticate with the secondary z/OSMF instance that is running in the sysplex where the software instance resides. In addition, if the primary z/OSMF instance must navigate an HTTP proxy server to connect with the secondary z/OSMF instance, you might also be required to authenticate with that HTTP proxy server. Therefore, you can need to specify the remote z/OSMF userid, password, and proxy userid and password.

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation that uses the z/OSMF Software Management task. That is, to export a software instance, the user ID must have READ access to the Software Management task and READ access to the SAF resources for the software instance being exported. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

Generating JCL to export a software instance is an asynchronous operation. Therefore, on completion of the initial POST request, the z/OSMF Software Management REST interface returns an HTTP response code of `202 Accepted` and a JSON document containing a URL for the status monitor for the request. The client performs GET requests to the supplied URL to monitor the status of the operation and to obtain the result set. For example:

```
{
 "statusurl":"url"
}
```

where:

statusurl
Indicates the URL that provides status for the software instance export request.

On subsequent GET requests to the status monitor URL:

- If the operation is not yet complete, an HTTP response code of `200 OK` is returned, along with a JSON document containing status information for the operation.
- If the operation has completed, then an HTTP response code of `200 OK` is returned, along with a JSON document containing status information and the desired result set.

- If the request has expired, then an HTTP response code of `404 Not found` is returned. That is, when the operation has completed, the z/OSMF server holds the result set for a finite length of time. After that time has passed, the result set is said to expire and will no longer be available for the client to obtain.

The response to GET requests on the status monitor URL includes the following JSON document:

```
{
 "status":"status",
 "percentcomplete":"percent",
 "jcl":["data-set-name(member-name)"]
}
```

status
The status of the export request. The status can be either “running” or “complete”.

percentcomplete
The percentage of the processing that is complete for this export software instance request, expressed as a whole number from 0 to100.

jcl
Ordered list of generated jobs for the export action. Each job is a unique member in the JCL data set. The values are fully qualified data set and member names of the form “data-set-name(member-name)”.

See “Error handling” on page 270 for the error response document containing a reason code, a list of one or more message strings to describe the errors that are detected during processing of a request, and for some errors, a stack trace of the exception.

Example

The following request generates an Export job for the software instance that is named *DB2V9* on *SYS123*.

```
POST /zosmf/swmgmt/swi/SYS123/DB2V9/export HTTP/1.1
Host: sys123.yourco.com
Content-Type: application/json
Accept-Language: en
{
"packagedir":"/u/userid/exportDir/",
"jcldataset":"USERID.SMJCL.CNTL",
"includedlibs":"yes",
"jobstatement":["//EXPORT   JOB (123456),'USER',NOTIFY=&SYSUID,",
"// MSGCLAS=H","//*"]
}
```

Figure 152. Sample request

A sample response is as follows:

```
HTTP/1.1 202 Accepted
Date: Tues, 21 November 2014 18:53:04 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"statusurl":"https:\/\/sys123.yourco.com\/zosmf\/swmgt\/statusmonitor\/export\/4837290198343"}
```

Figure 153. Sample response

The above response indicates the request to generate an Export job for the software instance has been accepted, and the status monitor URL is provided. A subsequent GET request to the status monitor URL is as follows:

```
GET /zosmf/swmgmt/statusmonitor/export/4837290198343 HTTP/1.1
Host: sys123.yourco.com
```

A sample response is as follows:

```
HTTP/1.1 200 OK
Date: Tues, 21 November 2014 18:53:19 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"status":"running", "percentcomplete":"65"}
```

Figure 154. Sample response

The above response indicates the operation to generate an Export job for the software instance is still running and 65% complete. A final request to the status monitor URL is as follows:

```
GET /zosmf/swmgmt/statusmonitor/export/4837290198343 HTTP/1.1
Host: sys123.yourco.com
```

A sample response is as follows:

```
HTTP/1.1 200 OK
Date: Tues, 21 November 2014 18:53:27 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"status":"complete", "percentcomplete":"100",
"jcl":["USERID.SMJCL.CNTL(IZUD01EX)"]
}
```

Figure 155. Sample response

Usage notes

The POST method to generate JCL that exports a software instance requires the subject software instance to be already defined to z/OSMF. If the software instance is not already defined, you can use the POST method to Add a new software instance to z/OSMF, followed by the PUT method to Load the products, features, and FMIDs for the new software instance if that software instance contains SMP/E managed software. Then, you can use the POST method to Export the software instance. For example:

Add the software instance:

```
POST /zosmf/swmgmt/swi HTTP/1.1
Host: sys123.yourco.com
Content-Type: application/json
Accept-Language: en
{
"name":"DB2V9",
"system":"sys123",
"description":"DB2",
"globalzone":"DB2.GLOBAL.CSI",
"targetzones":["DB2TGT"],
}
```

Load the products, features, and FMIDs for the software instance:

```
PUT /zosmf/swmgmt/swi/sys123/DB2/products HTTP/1.1
Host: sys123.yourco.com
```

Create JCL to export the software instance:

```
POST /zosmf/swmgmt/swi/sys123/DB2/export HTTP/1.1
Host: sys123.yourco.com
Content-Type: application/json
Accept-Language: en
{
```

```
"packagedir":"/u/userid/exportDir/",
"jcldataset":"USERID.SMJCL.CNTL",
"includedlibs":"yes",
,
"jobstatement":["//EXPORT JOB (123456),'USER',NOTIFY=&SYSUID,",
"// MSGCLAS=H","//*"]
}
```

The response from the POST method to create JCL to export the software instance specifies the members in the JCL data set that contain the generated jobs to perform the export action. You can use the z/OS jobs REST interface to submit and obtain the status of the export job. For example:

Submit the export job:

```
PUT /zosmf/restjobs/jobs HTTP/1.1
Host: sys123.yourco.com
Content-Type: application/json
{
"file":"//'USERID.SMJCL.CNTL(IZUD01EX)'"
}
```

The response to this PUT method provides the job ID and the job name for the submitted job.

Obtain status for the export job:

```
GET /zosmf/restjobs/jobs/IZUD01EX/job-id HTTP/1.1
Host: sys123.yourco.com
```

Sample REXX exec

A sample REXX exec that is named IZUDXEXP is provided in the SYS1.SAMPLIB data set to illustrate a program that uses Software management services to do the following:

- Add a software instance.
- Load the SMP/E Products, Features, and FMIDs for the software instance.
- Generate JCL to export the software instance.
- Run the generated JCL to create a portable software instance.

The sample REXX exec uses the HTTP REXX client of the z/OS Web Enablement Toolkit to perform HTTP operations to the z/OSMF server.

Modify the properties of a software instance

You can use this operation to modify the properties of a software instance that is defined in z/OSMF, including changing the global zone, target zones, or non-SMP/E managed data sets associated with the software instance. The modify operation updates only the definition of the software instance in z/OSMF. The physical data sets that compose the software instance are not affected.

HTTP method and URI path

```
PUT /zosmf/swmgmt/swi/<system-nickname>/<swi-name>
```

---

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.
- ***<system-nickname>/<swi-name>*** further qualifies the request and indicates the specific software instance to be modified. A software instance is uniquely identified by its name (*swi-name*) and the nickname (*system-nickname*) of the z/OSMF host system that has access to the volumes and data sets where the software instance resides.

  To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

Standard headers

Use the following standard HTTP header with this request:

Content-Type
: Identifies the type of input content provided by the caller. The JSON content type ("Content-Type: application/json") is used for the JSON document included as input with this request.

Accept-Language
: Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, English is used.

Custom headers

None.

Request content

Your request must include a JSON object that describes all the properties of the software instance to be modified, for example:

```
{
  "name":"swi-name",
  "system":"system-nickname",
  "description":"swi-description",
  "globalzone":"global-zone",
  "targetzones":["target-zones"],
  "categories":"["categories"],
  "datasets":[
  {
    "dsname":"data-set-name",
    "volume":"volume-serial"
  }]

}
```

Figure 156. Adding a software instance: request content

where:

swi-name
Name of the software instance. The name can contain up to 30 non-blank characters, including alphanumeric characters (A-Z, a-z, and 0-9), mathematical symbols (< > - = | \), punctuation marks (? ! : ' " /), and special characters ($ _ # @ ^). The name is required and must be unique on the system.

system-nickname
Nickname of the system that has access to the volumes and data sets where the software instance resides. Use the nickname that is specified for the system definition in the z/OSMF Systems task. The nickname is required.

To manage the systems defined to z/OSMF, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

swi-description
Description of the software instance. The description is optional and can contain a maximum of 256 characters.

global-zone
CSI data set that contains the global zone used to manage the software. If specified, must comply with z/OS data set naming conventions, and must end with *.CSI*.

target-zones
Comma-separated list of the target zones to be included in the software instance. A list of 0 or more zone names, each target zone name must be 1-7 characters long; the valid characters are uppercase alphabetic characters (A-Z), numeric characters (0-9), and special characters (@ # $). The first character must be an alphabetic character.

categories
Comma-separated list of the categories to which the software instance is assigned. Each category name can contain up to 30 non-blank characters, including alphanumeric characters (A-Z, a-z, and 0-9), mathematical symbols (< > - = | \), punctuation marks (? ! : ' " /), and special characters ($ _ # @ ^). Assigning the software instance to a category is optional.

datasets
Array that contains each non-SMP/E managed data set to be added to the software instance. Adding non-SMP/E managed data sets to the software instance is optional.

data-set-name
Name of the non-SMP/E managed data set. You cannot specify data set members or a subset of a data set. A data set name is required if you are adding non-SMP/E managed data sets to the software instance. The data set name must comply with z/OS data set naming conventions

volume-serial
Volume on which the non-SMP/E managed data set resides. The volume serial is required if the data set is not cataloged, and the volume must be accessible by the system where the software instance resides. The volume serial must be 6 characters long; the valid characters are uppercase alphabetic characters (A-Z) and numeric characters (0-9).

When modifying a software instance, the software management REST interface validates the information that is supplied in the JSON object before completing the request. All input values must be valid and syntactically correct. If any errors are discovered, the modify request fails and appropriate messages are returned.

The name and system, properties are required for every software instance. If either of these are not specified or are specified as null, then the request finishes with HTTP response code 400 (Badrequest) and reason code 4 (The required property {name|system} is missing. Add the missing property and retry the request.).

If the globalzone property is specified, then the targetzones property is required. Therefore, if globalzone is specified and targetzones is not, then the request finishes with HTTP response code400 (Bad request) and reason code 4 (The required property {targetzones} is missing. Add the missing property and retry the request.).

If the globalzone property is not specified, then the datasets property is required. That is, the software instance must contain either SMP/E managed software, or at least one non-SMP/Emanaged data set. Therefore, if both globalzone and datasets are not specified or are specified as null, then the request finishes with HTTP response code 400 (Bad request) and reason code 42 (Both of the properties {globalzone} and {datasets} are missing, but one is required. Add one or both of the missing properties and retry the request.).

The targetzone property can only be specified if the globalzone property is also specified. Therefore, if targetzone is specified but globalzone is not, then the request finishes with HTTP response code 400 (Bad request) and reason code 43 (The {targetzone} property requires the{globalzone} property, but the {globalzone} property is missing. Either add the {globalzone}property or remove the {targetzone} property and retry the request.)

If the input values are valid, the software instance that is defined in this request overwrites the existing software instance definition. In a modify request, you must supply all the existing properties for the software instance, and modify only those properties you want to update. To delete a property, specify all the existing properties for the software instance in the JSON object and either exclude the property to be deleted or set its value to *null*.

Usage considerations

The SMP/E global zone and target zones define the installed products, features, and FMIDs for a software instance. If you modify these properties for a software instance, perform a PUT method to load the products, features, and FMIDs for the updated software instance. Doing so refreshes the product, feature, and FMID information known to z/OSMF for the subject software instance. For instructions, see "Load the products, features, and FMIDs for a software instance" on page 305.

For other usage considerations, see "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

To submit a modify software instance request through the software management services, the user ID initiating the request requires the same authorizations as when performing a modify operation using the z/OSMF Software Management task. That is, the user ID must have READ access to the Software

Management task. The user ID must also have either CONTROL or UPDATE access to the SAF resources corresponding to the software instance to be modified, as follows:

- The name, system, and categories properties are used to create the SAF resource names for the software instance; therefore, to modify any of these properties, the user ID must have CONTROL access to the existing SAF resource names and to the new SAF resource names.
- are implicitly added to z/OSMF through the modify operation, the user ID must have CONTROL access to the SAF resources corresponding to the specified categories.
- To modify any other property, the user ID must have UPDATE access.

For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 270.

Example

In the following example, the PUT method is used to modify the description property for software instance *DB2V9* on system *PEV174*.

```
PUT /zosmf/swmgmt/swi/PEV174/DB2V9 HTTP/1.1
Host: pev174.yourco.com

Content-Type: application/json
Accept-Language: en
{
  "name":"DB2V9",
  "system":"PEV174",
  "description":"DB2 new description",
  "globalzone":"DB2.GLOBAL.CSI",
  "targetzones":["DB2TGT"],
  "datasets":[
  {"dsname":"USER.DB2V9.PROCLIB","volume": "LV1234"},
  {"dsname":"USER.DB2V9.SAMPLES","volume":"LV1234"}]
}
```

Figure 157. Sample request to modify a software instance

Figure 158 provides a sample response, indicating that the update was successful.

```
HTTP/1.1 200 OK
Date: Tues, 22 July 2014 18:53:27 +00006GMT
```

Figure 158. Sample response for a modify software instance request

Load the products, features, and FMIDs for a software instance

You can use this operation to analyze the SMP/E global zone and target zones for a software instance to identify the installed products, features, and FMIDs in the instance, and to load that information into the z/OSMF Software Management task database.

HTTP method and URI path

```
PUT /zosmf/swmgmt/swi/<system-nickname>/<swi-name>/products
```

---

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.
- ***<system-nickname>/<swi-name>*** further qualifies the request and indicates the specific software instance to be retrieved. A software instance is uniquely identified by its name (*swi-name*) and the nickname (*system-nickname*) of the z/OSMF host system that has access to the volumes and data sets where the software instance resides.

  To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.
- **products** indicates that the products, features, and FMIDs included in the software instance are to be obtained and loaded into the Software Management task database.

When you issue this request, z/OSMF searches the SMP/E global zone and target zones associated with the software instance and gathers information about the software instance and its products, features, and FMIDs. z/OSMF returns this information in a JSON object, and stores this information in the Software Management task database.

Standard headers

Use the following standard HTTP header with this request:

Content-Type
: Identifies the type of input content provided by the caller. Use the JSON content type ("Content-Type: application/json") if a JSON document is included as input with this request.

Accept-Language
: Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, English is used.

Custom headers

None.

Request content

If the software instance does not reside in the same sysplex as the primary z/OSMF instance, you might be required to authenticate with the secondary z/OSMF instance that is running in the sysplex where the software instance resides. In addition, if the primary z/OSMF instance must navigate an HTTP proxy server to connect with the secondary z/OSMF instance, you might also be required to authenticate with that HTTP proxy server. To do so, include the following JSON object in your request:

```
{
  "zosmfuid":"zosmf-user-ID",
  "zosmfpw":"zosmf-password",
  "proxyuid":"proxy-user-ID",
  "proxypw":"proxy-password"
}
```

Figure 159. Request content to authenticate with a secondary z/OSMF instance and an HTTP proxy server

where:

zosmf-user-ID
User ID for authenticating with the secondary z/OSMF instance.

zosmf-password
Password for authenticating with the secondary z/OSMF instance.

proxy-user-ID
User ID for authenticating with the HTTP proxy server.

proxy-password
Password for authenticating with the HTTP proxy server.

Include the JSON object in the request only if you are required to authenticate with a secondary z/OSMF instance or an HTTP proxy server. Otherwise, omit the JSON object.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

To submit requests through the software management services, the user ID initiating the request requires the same authorizations as when performing an analogous operation using the z/OSMF Software Management task. That is, to retrieve the product, feature, and FMID information, the user ID must have READ access to the Software Management task and UPDATE access to the SAF resources for the software instance being updated. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request was accepted. If the request was accepted, the service returns status code `202 Accepted` and a JSON object that contains a URL (`{"statusurl":"url"}`). To monitor the status of the retrieve product, feature, and FMID information request and to obtain the results, perform GET requests to the supplied URL. Only the user ID that initiates the retrieve product, feature, and FMID information request is authorized to check the status and obtain the results. One of the following responses is returned from the get status request:

- If the retrieve product, feature, and FMID information request is still in progress, an HTTP response code of `200 OK` is returned, along with the following JSON object: `{"status":"status"}`.
- If the retrieve product, feature, and FMID information request is complete, an HTTP response code of `200 OK` is returned, along with the following JSON object:

```
{
  "status":"status",
  "swi":{
    "name":"swi-name",
    "system":"system-nickname",
    "description":"swi-description"
```

```
    "globalzone":"global-zone-name",
    "targetzones":["target-zone-name"],
    "categories":["category-name"],
    "productinforetrieved":"date-retrieved",
    "lastmodified":"date-modified",
    "modifiedby":"modified-user-ID",
    "created":"date-created",
    "createdby":"created-user-ID",
    "locked":"date-locked",
    "lockedby":"locked-user-ID",
    "datasets":[{
      "dsname":"data-set-name",
      "volume":"volume-serial"
    }],
    "products":[{
      "prodname":"product-name",
      "prodid":"product-ID",
      "release":"version-release-modification",
      "vendor":"vendor-name",
      "generalavailability":"general-availability-date",
      "endofservice":"end-of-service-date",
      "url":"product-URL",
      "productinfofileversion":"product-file-version",
      "features":[{
        "feature":"feature-name",
        "fmids":[{
          "fmid":"fmid-name",
          "description":"fmid-description",
          "targetzones":["fmid-target-zone-name"]
        }]
      }]
    }]
  }
}
```

where:

status Status of the retrieve product, feature, and FMID information request. The status is either *running* or *complete*.

swi JSON object to describe a software instance.

swi-name
Name of the software instance.

system-nickname
Nickname of the z/OSMF host system that has access to the global zone CSI data set included in the software instance. To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

swi-description
Description of the software instance.

global-zone-name
Name of the CSI data set that contains the global zone used to manage the software.

target-zone-name
Array of the target zones in the specified global zone that describe the software.

category-name
Array of the categories to which the software instance is assigned.

date-retrieved
Date and time in ISO 8601 format that the product, feature, and FMID information for the software instance was last retrieved from the CSI data set. For example, 2014-08-20T19:23:25+00:00Z.

date-modified
Date and time in ISO 8601 format that the software instance was last modified. For example, 2014-08-20T19:23:25+00:00Z.

modified-user-ID
User ID of the user who last modified the software instance.

date-created
Date and time in ISO 8601 format that the software instance was created. For example, 2014-08-20T19:23:25+00:00Z.

created-user-ID
User ID of the user who created the software instance.

date-locked
Date and time in ISO 8601 format that the software instance was locked for an impending update. For example, 2014-08-20T19:23:25+00:00Z. If null, the software instance is not locked.

locked-user-ID
User ID of the user who locked the software instance.

datasets
Array of the non-SMP/E managed data sets that are included in the software instance.

data-set-name
Name of the data set.

volume-serial
Volume serial for the volume where the data set resides.

products
Array of the products included in the software instance.

product-name
Name of the product.

product-ID
Identifier for the product.

version-release-modification
Version, release, and modification level of the product. The value has the format *VV.RR.MM*, where *VV* is the two-digit version, *RR* is the two-digit release, and *MM* is the two-digit modification level.

vendor-name
Name of the vendor that provides the product.

general-availability-date
Date and time in ISO 8601 format that a version or release of the product became available to all users. For example, 2014-08-20T19:23:25+00:00Z.

end-of-service-date
Date and time in ISO 8601 format that service support ends for the product. For example, 2014-08-20T19:23:25+00:00Z.

product-URL
URL that links to additional information about the product. This information can include, for example, product life cycle dates, product highlights, planning information, and technical descriptions.

product-file-version
Version of the most recent product information file that supplied information about the product. The version represents the date and time in ISO 8601 format that file was created or last updated. For example, 2014-08-20T19:23:25+00:00Z.

features
Array of the features contained in the software instance.

feature-name
Name of the feature.

fmids Array of the FMIDs contained in the software instance.

fmid-name
Name of the FMID.

fmid-description
Description of the FMID.

fmid-target-zone-name
Array of the target zones where the FMID is installed.

- If the retrieve product, feature, and FMID information request is complete but the results are no longer available, an HTTP response code of `404 Not found` is returned. z/OSMF makes the results available for a client application for a finite period of time. When that time elapses, the results are no longer available; in which case, the client must reissue the request.
- If the identified software instance has no global zone property, then products, features, and FMIDs cannot be loaded for this software instance. If such a software instance is the target of such a request, an HTTP response code 400 (Bad request) with reason code 41 will be returned (The request cannot be performed because the software instance does not have a global zone.).

If the retrieve product, feature, and FMID information request cannot be processed, a status code of 4*nn* or 5*nn* is returned, indicating that an error has occurred. For more details, see "Error handling" on page 270.

Example

In the following example, the PUT method is used to retrieve the product, feature, and FMID information for software instance *DB2V9* on system *SYS123*.

```
PUT /zosmf/swmgmt/swi/SYS123/DB2V9/products HTTP/1.1
Host: sys123.yourco.com
```

Figure 160. Sample request to retrieve the product, feature, and FMID information for a software instance

Figure 161 provides a sample response, indicating that the retrieve product, feature, and FMID information request has been accepted and supplying the URL to use for monitoring the status of that request.

```
HTTP/1.1 202 Accepted
Date: Tues, 21 November 2014 18:53:04 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"statusurl":"https:\/\/sys123.yourco.com\/zosmf\/swmgt\/statusmonitor\/prodload
\/4837290198343"}
```

Figure 161. Sample response for a retrieve product, feature, and FMID information request

To check the status of the retrieve product, feature, and FMID information request, submit the following request:

```
GET /zosmf/swmgmt/statusmonitor/prodload/4837290198343 HTTP/1.1
Host: sys123.yourco.com
```

Figure 162. Sample request to obtain the status of a retrieve product, feature, and FMID information request

Figure 163 provides a sample get status response, indicating that the retrieve product, feature, and FMID information request is in progress.

```
HTTP/1.1 200 OK
Date: Tues, 21 November 2014 18:53:19 +00005GMT
Content-Type: application/json
Content-Language: en
Connection: close
{"status":"running"}
```

Figure 163. Sample get status response when the retrieve product, feature, and FMID information request is in progress

Figure 164 provides a sample get status response, indicating that the retrieve product, feature, and FMID information request is complete.

```
HTTP/1.1 200 OK
Date: Tues, 21 November 2014 18:53:36 +00006GMT
Content-Type: application/json
Content-Language: en
Connection: close
{
"status":"complete", "swi":{
"name":"DB2V9", "system":"PEV174", "description":null,
"globalzone":"DB2.GLOBAL.CSI", "targetzones":["DB2TGT"], "categories":null,
"productinforetrieved":"2014-08-20T19:23:25Z", "lastmodified":"2014-08-
20T19:23:25Z", "modifiedby":"FRED", "created":"2014-08-20T19:23:25Z",
"createdby":"BARNEY", "locked":null, "lockedby":null, "datasets":
[{"dsname":"USER.DB2V9.PROCLIB", "volume": "LV1234"},
{"dsname":"USER.DB2V9.SAMPLES", "volume":"LV1234"}], "products":
[{"prodname":"DB2 for z/OS", "prodid":"5635-DB2", "release":"09.01.00",
"vendor":"IBM", "generalavailability":"20006-06-09T19:23:25Z",
"endofservice":"2014-06-27T19:23:25Z", "url":null,
"productinfofileversion":"2014-01-01", "features":[{"feature":"DB2 Base",
"fmids":[{"fmid":"HDB9910", "description":"DB2 BASE/TSO", "targetzones":
["DB2V9T"]}]}]}
}
```

Figure 164. Sample get status response when the retrieve product, feature, and FMID information request is complete

Delete a software instance

You can use this operation to remove a software instance definition from z/OSMF. The delete operation removes only the definition of the software instance from z/OSMF. The physical data sets that compose the software instance are not affected.

HTTP method and URI path

```
DELETE /zosmf/swmgmt/swi/<system-nickname>/<swi-name>
```

---

where:

- **zosmf/swmgmt** identifies the software management services.
- **swi** informs the service that the request is for the software instance object.
- ***<system-nickname>/<swi-name>*** further qualifies the request and indicates the specific software instance to be deleted. A software instance is uniquely identified by its name (*swi-name*) and the nickname (*system-nickname*) of the z/OSMF host system that has access to the volumes and data sets where the software instance resides.

  To obtain information about the specified system, you can use the z/OSMF topology services. For more details, see "Topology services" on page 313.

Standard headers

Use the following standard HTTP header with this request:

Accept-Language
: Identifies the preferred language for messages that may be returned to the caller. Acceptable values are "Accept-Language: en" (English) and "Accept-Language: ja" (Japanese). Any other language value is ignored and English is used instead. In addition, if the header is not specified, English is used.

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

To submit a delete software instance request through the software management services, the user ID initiating the request requires the same authorizations as when performing a remove operation using the z/OSMF Software Management task. That is, the user ID must have READ access to the Software Management task, and CONTROL access to the SAF resources corresponding to the software instance to be deleted. For information about access controls for the Software Management task, see *IBM z/OS Management Facility Configuration Guide*.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 270.

Example

In the following example, the DELETE method is used to delete software instance *DB2V9* on system *PEV174*.

```
DELETE /zosmf/swmgmt/swi/PEV174/DB2V9 HTTP/1.1
Host: pev174.yourco.com
```

Figure 165. Sample request to delete a software instance

Figure 166 provides a sample response, indicating that the delete operation was successful.

```
HTTP/1.1 200 OK
Date: Tues, 22 July 2014 18:53:27 +00006GMT
```

Figure 166. Sample response for a delete software instance request

Topology services

The topology services is an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. A set of REST services is provided for working with the groups, sysplexes, central processor complexes (CPCs), and systems that are defined to z/OSMF, as described in this topic.

Table 194 lists the operations that the topology services provide.

Table 194. Operations provided through the topology services.

Operation	HTTP method and URI path
"List the systems defined to z/OSMF" on page 315	GET /zosmf/resttopology/systems
"List the groups defined to z/OSMF" on page 318	GET /zosmf/resttopology/groups
"List the systems included in a group" on page 320	GET /zosmf/resttopology/systems/groupName/<*groupName*>
"List the sysplexes defined to z/OSMF" on page 323	GET /zosmf/resttopology/sysplexes
"List the systems included in a sysplex" on page 325	GET /zosmf/resttopology/systems/sysplexName/<*sysplexName*>
"List the systems included in a CPC" on page 328	GET /zosmf/resttopology/systems/cpcName/<*cpcName*>

Required authorizations

To submit requests through the topology services, your user ID requires authorization to the Systems task provided in z/OSMF. Ensure that your user ID has READ access to the following resource profile in the ZMFAPLA class: <SAF-prefix>.ZOSMF.SETTINGS.SYSTEMS.VIEW. By default, users with user IDs connected to the IZUADMIN and IZUUSER security groups can access the topology services.

For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP response data

The JSON content type (`"Content-Type: application/json"`) is used for response data. The following JSON object is used by all topology services for returning data about the requested operation:

```
{
 "items": "item-list",
 "numRows": "total-items"
}
```

where:

item-list
: Array that contains the items that were retrieved. The attributes provided in the array depend on the requested operation.

total-items
Number of items retrieved.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains a message that describes the error.

The following HTTP status codes are valid:

HTTP 200 OK
Success.

HTTP 400 Bad request
The request contained incorrect parameters.

HTTP 401 Unauthorized
The submitter of the request did not authenticate to z/OSMF or is not authorized to use the topology services.

HTTP 403 Forbidden
The server rejected the request.

HTTP 404 Bad URL
The target of the request (a URL) was not found.

HTTP 405 Method not allowed
The service does not support the HTTP method specified for the request.

HTTP 500 Internal server error
A programming error occurred.

Error logging

Errors from the topology services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

List the systems defined to z/OSMF

You can use this operation to obtain a list of the systems that are defined to a z/OSMF instance.

HTTP method and URI path

```
GET /zosmf/resttopology/systems
```

where:

- **zosmf/resttopology** identifies the topology services.
- **systems** informs the service that the request is to retrieve a list of the systems that are defined to the z/OSMF instance.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 313.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 314.

If the request was successful, the response also includes the following JSON object:

```
{
 "items":[
   {
    "systemNickName":"system-nickname",
    "systemName":"system-name",
    "sysplexName":"sysplex-name",
    "groupNames":"group-names",
    "url":"url",
    "zosVR":"zos-level",
    "jesMemberName":"JES-member-name",
    "jesType":"JES-type",
    "cpcName":"CPC-name",
    "cpcSerial":"CPC-serial",
    "httpProxyName":"proxy-name",
```

```
    "ftpDestinationName":"server-name"
   }
 ],
 "numRows":"total-items"
}
```

where:

system-nickname
Unique name assigned to the system definition.

system-name
Name specified for the system on the SYSNAME parameter in the IEASYSxx parmlib member.

sysplex-name
Name of the sysplex where the z/OS system is a member. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.

group-names
Comma-separated list of the groups to which the system is assigned.

url URL used to access the z/OSMF instance that resides in the same sysplex as the system identified by the **systemName** attribute. Or, the URL used to access the application server that is hosting the server-side code for the plug-ins your enterprise imported into z/OSMF.

zos-level
Version and release of the z/OS image installed on the system. The version and release has the format *z/OS VxxRyy* where *V* stands for version, *xx* is the version number, *R* stands for release, and *yy* is the release number. For example, *z/OS V2R1*.

JES-member-name
JES2 multi-access spool (MAS) member name or JES3 complex member name that is assigned to the primary job entry subsystem (JES) that is running on the system.

JES-type
Type for the primary job entry subsystem running on the system. The type is either JES2 or JES3.

CPC-name
Name specified for the central processor complex (CPC) at the support element (SE) of that processor complex.

CPC-serial
Serial number of the CPC.

proxy-name
Name of the HTTP proxy definition that specifies the settings required to access the system through an HTTP or SOCKS proxy server.

server-name
Name of the server definition that specifies the settings required to access the FTP or SFTP server that is running on the system.

total-items
Number of system definitions that were retrieved.

Example

In the following example, the GET method is used to retrieve a list of the systems that are defined to the z/OSMF instance that has a host name of *zosmf1.yourco.com*.

```
GET /zosmf/resttopology/systems HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 167. Sample request to retrieve a list of systems

A sample response is shown in Figure 168.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
 "items":[
   {
    "systemNickName":"sys1",
    "systemName":"sys1",
    "sysplexName":"plex1",
    "groupNames":"test,development",
    "url":"https://zosmf1.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY1",
    "jesType":"JES2",
    "cpcName":"",
    "cpcSerial":"",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   },
   {
    "systemNickName":"sys2",
    "systemName":"sys2",
    "sysplexName":"plex2",
    "groupNames":"production",
    "url":"https://zosmf2.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY2",
    "jesType":"JES3",
    "cpcName":"",
    "cpcSerial":"",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs-sftp"
   },
   {
    "systemNickName":"sys3",
    "systemName":"sys3",
    "sysplexName":"plex3",
    "groupNames":"test",
    "url":"https://zosmf3.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY3",
    "jesType":"JES2",
    "cpcName":"",
    "cpcSerial":"",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   }
 ],
 "numRows":"3"
}
```

Figure 168. Sample response from a request to retrieve a list of systems

List the groups defined to z/OSMF

You can use this operation to obtain a list of the groups that are defined to a z/OSMF instance.

HTTP method and URI path

```
GET /zosmf/resttopology/groups
```

where:

- **zosmf/resttopology** identifies the topology services.
- **groups** informs the service that the request is to retrieve a list of the groups that are defined to the z/OSMF instance.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 313.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 314.

If the request was successful, the response also includes the following JSON object:

```
{
 "items":[
   {
    "name":"group-name",
    "description":"group-description",
    "systemNickNames":"system-nicknames"
   }
 ],
 "numRows":"total-items"
}
```

where:

group-name
Name of the groups defined to z/OSMF. A value of <*None*> indicates that one or more systems are not assigned to a group.

group-description
Description of the group.

system-nicknames
Comma-separated list of the systems assigned to the group. Each system is identified by its nickname.

total-items
Number of groups that were retrieved.

Example

In the following example, the GET method is used to retrieve a list of the groups that are defined to the z/OSMF instance that has a host name of *zosmf1.yourco.com*.

```
GET /zosmf/resttopology/groups HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 169. Sample request to retrieve a list of groups

A sample response is shown in Figure 170.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
 "items":[
   {
    "name":"development",
    "description":"This group contains the systems used by development.",
    "systemNickNames":"sys1"
   },
   {
    "name":"production",
    "description":"This group contains the systems that are in production.",
    "systemNickNames":"sys2"
   },
   {
    "name":"test",
    "description":"This group contains the systems that are used for testing code.",
    "systemNickNames":"sys1,sys3"
   }
 ],
 "numRows":"3"
}
```

Figure 170. Sample response from a request to retrieve a list of groups

List the systems included in a group

You can use this operation to obtain a list of the systems that are included in a group.

HTTP method and URI path

```
GET /zosmf/resttopology/systems/groupName/<groupName>
```

---

where:

- **zosmf/resttopology** identifies the topology services.
- **systems/groupName** informs the service that the request is to retrieve a list of the systems that are defined to a specific group.
- **<groupName>** identifies the group for which to obtain the list of systems. If the group name contains a number sign (#), encode the number sign as *%23*. For example, if the group name is *test#systems*, specify *test%23systems*. Otherwise, the service will truncate *#systems*, and use *test* as the group name.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 313.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 314.

If the request was successful, the response also includes the following JSON object:

```
{
 "items":[
   {
    "systemNickName":"system-nickname",
    "systemName":"system-name",
    "sysplexName":"sysplex-name",
    "groupNames":"group-names",
    "url":"url",
    "zosVR":"zos-level",
    "jesMemberName":"JES-member-name",
```

```
    "jesType":"JES-type",
    "cpcName":"CPC-name",
    "cpcSerial":"CPC-serial",
    "httpProxyName":"proxy-name",
    "ftpDestinationName":"server-name"
   }
 ],
 "numRows":"total-items"
}
```

where:

system-nickname
Unique name assigned to the system definition.

system-name
Name specified for the system on the SYSNAME parameter in the IEASYSxx parmlib member.

sysplex-name
Name of the sysplex where the z/OS system is a member. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.

group-names
Comma-separated list of the groups to which the system is assigned.

url URL used to access the z/OSMF instance that resides in the same sysplex as the system identified by the **systemName** attribute. Or, the URL used to access the application server that is hosting the server-side code for the plug-ins your enterprise imported into z/OSMF.

zos-level
Version and release of the z/OS image installed on the system. The version and release has the format *z/OS VxxRyy* where *V* stands for version, *xx* is the version number, *R* stands for release, and *yy* is the release number. For example, *z/OS V2R1*.

JES-member-name
JES2 multi-access spool (MAS) member name or JES3 complex member name that is assigned to the primary job entry subsystem (JES) that is running on the system.

JES-type
Type for the primary job entry subsystem running on the system. The type is either JES2 or JES3.

CPC-name
Name specified for the central processor complex (CPC) at the support element (SE) of that processor complex.

CPC-serial
Serial number of the CPC.

proxy-name
Name of the HTTP proxy definition that specifies the settings required to access the system through an HTTP or SOCKS proxy server.

server-name
Name of the server definition that specifies the settings required to access the FTP or SFTP server that is running on the system.

total-items
Number of system definitions that were retrieved.

Example

In the following example, the GET method is used to retrieve a list of the systems that are defined to the z/OSMF instance with host name *zosmf1.yourco.com* and that are assigned to the group *test*.

```
GET /zosmf/resttopology/systems/groupName/test HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 171. Sample request to retrieve a list of systems included in a group

A sample response is shown in Figure 172.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
 "items":[
   {
    "systemNickName":"sys1",
    "systemName":"sys1",
    "sysplexName":"plex1",
    "groupNames":"test,development",
    "url":"https://zosmf1.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY1",
    "jesType":"JES2",
    "cpcName":"",
    "cpcSerial":"",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   },
   {
    "systemNickName":"sys3",
    "systemName":"sys3",
    "sysplexName":"plex3",
    "groupNames":"test",
    "url":"https://zosmf3.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY3",
    "jesType":"JES2",
    "cpcName":"",
    "cpcSerial":"",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   }
 ],
 "numRows":"2"
}
```

Figure 172. Sample response from a request to retrieve a list of systems included in a group

List the sysplexes defined to z/OSMF

You can use this operation to obtain a list of the sysplexes that are defined to a z/OSMF instance.

HTTP method and URI path

```
GET /zosmf/resttopology/sysplexes
```

where:

- **zosmf/resttopology** identifies the topology services.
- **sysplexes** informs the service that the request is to retrieve a list of the sysplexes that are defined to the z/OSMF instance.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 313.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 314.

If the request was successful, the response also includes the following JSON object:

```
{
 "items":[
   {
    "sysplexName":"sysplex-name",
    "systemNickNames":"system-nicknames"
   }
 ],
 "numRows":"total-items"
}
```

where:

sysplex-name
Name of the sysplex where the z/OS system is a member. The name is the value specified for the

SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility. A value of <*Not Specified*> indicates that one or more systems are not assigned to a sysplex.

system-nicknames
Comma-separated list of the systems assigned to the sysplex. Each system is identified by its nickname.

total-items
Number of sysplexes that were retrieved.

Example

In the following example, the GET method is used to retrieve a list of the sysplexes that are defined to the z/OSMF instance that has a host name of *zosmf1.yourco.com*.

```
GET /zosmf/resttopology/sysplexes HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 173. Sample request to retrieve a list of sysplexes

A sample response is shown in Figure 174.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
 "items":[
   {
    "sysplexName":"plex1",
    "systemNickNames":"sys1"
   },
   {
    "sysplexName":"plex2",
    "systemNickNames":"sys2"
   },
   {
    "sysplexName":"plex3",
    "systemNickNames":"sys3"
   }
 ],
 "numRows":"3"
}
```

Figure 174. Sample response from a request to retrieve a list of sysplexes

List the systems included in a sysplex

You can use this operation to obtain a list of the systems that are included in a sysplex.

HTTP method and URI path

```
GET /zosmf/resttopology/systems/sysplexName/<sysplexName>
```

where:

- **zosmf/resttopology** identifies the topology services.
- **systems/sysplexName** informs the service that the request is to retrieve a list of the systems that are included in a specific sysplex.
- **<sysplexName>** identifies the sysplex for which to obtain the list of systems.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 313.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 314.

If the request was successful, the response also includes the following JSON object:

```
{
 "items":[
   {
    "systemNickName":"system-nickname",
    "systemName":"system-name",
    "sysplexName":"sysplex-name",
    "groupNames":"group-names",
    "url":"url",
    "zosVR":"zos-level",
    "jesMemberName":"JES-member-name",
    "jesType":"JES-type",
    "cpcName":"CPC-name",
    "cpcSerial":"CPC-serial",
```

```
    "httpProxyName":"proxy-name",
    "ftpDestinationName":"server-name"
   }
 ],
 "numRows":"total-items"
}
```

where:

system-nickname
Unique name assigned to the system definition.

system-name
Name specified for the system on the SYSNAME parameter in the IEASYSxx parmlib member.

sysplex-name
Name of the sysplex where the z/OS system is a member. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.

group-names
Comma-separated list of the groups to which the system is assigned.

url URL used to access the z/OSMF instance that resides in the same sysplex as the system identified by the **systemName** attribute. Or, the URL used to access the application server that is hosting the server-side code for the plug-ins your enterprise imported into z/OSMF.

zos-level
Version and release of the z/OS image installed on the system. The version and release has the format *z/OS VxxRyy* where *V* stands for version, *xx* is the version number, *R* stands for release, and *yy* is the release number. For example, *z/OS V2R1*.

JES-member-name
JES2 multi-access spool (MAS) member name or JES3 complex member name that is assigned to the primary job entry subsystem (JES) that is running on the system.

JES-type
Type for the primary job entry subsystem running on the system. The type is either JES2 or JES3.

CPC-name
Name specified for the central processor complex (CPC) at the support element (SE) of that processor complex.

CPC-serial
Serial number of the CPC.

proxy-name
Name of the HTTP proxy definition that specifies the settings required to access the system through an HTTP or SOCKS proxy server.

server-name
Name of the server definition that specifies the settings required to access the FTP or SFTP server that is running on the system.

total-items
Number of system definitions that were retrieved.

Example

In the following example, the GET method is used to retrieve a list of the systems that are defined to the z/OSMF instance with host name *zosmf1.yourco.com* and that are included in sysplex *plex1*.

```
GET /zosmf/resttopology/systems/sysplexName/plex1 HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 175. Sample request to retrieve a list of systems included in a sysplex

A sample response is shown in Figure 176.

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 05:39:28 +0000GMT
Connection: close

{
 "items":[
   {
    "systemNickName":"sys1",
    "systemName":"sys1",
    "sysplexName":"plex1",
    "groupNames":"test,development",
    "url":"https://zosmf1.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY1",
    "jesType":"JES2",
    "cpcName":"",
    "cpcSerial":"",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   }
 ],
 "numRows":"1"
}
```

Figure 176. Sample response from a request to retrieve a list of systems included in a sysplex

List the systems included in a CPC

You can use this operation to obtain a list of the systems that are included in a central processor complex (CPC).

HTTP method and URI path

```
GET /zosmf/resttopology/systems/cpcName/<cpcName>
```

---

where:

- **zosmf/resttopology** identifies the topology services.
- **systems/cpcName** informs the service that the request is to retrieve a list of the systems that are included in a specific CPC.
- **<cpcName>** identifies the CPC for which to obtain the list of systems.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 313.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 314.

If the request was successful, the response also includes the following JSON object:

```
{
 "items":[
   {
    "systemNickName":"system-nickname",
    "systemName":"system-name",
    "sysplexName":"sysplex-name",
    "groupNames":"group-names",
    "url":"url",
    "zosVR":"zos-level",
    "jesMemberName":"JES-member-name",
    "jesType":"JES-type",
    "cpcName":"CPC-name",
```

```
    "cpcSerial":"CPC-serial",
    "httpProxyName":"proxy-name",
    "ftpDestinationName":"server-name"
   }
 ],
 "numRows":"total-items"
}
```

where:

system-nickname
Unique name assigned to the system definition.

system-name
Name specified for the system on the SYSNAME parameter in the IEASYSxx parmlib member.

sysplex-name
Name of the sysplex where the z/OS system is a member. The name is the value specified for the SYSPLEX parameter of the cross-system coupling facility (XCF) couple data set format utility.

group-names
Comma-separated list of the groups to which the system is assigned.

url URL used to access the z/OSMF instance that resides in the same sysplex as the system identified by the **systemName** attribute. Or, the URL used to access the application server that is hosting the server-side code for the plug-ins your enterprise imported into z/OSMF.

zos-level
Version and release of the z/OS image installed on the system. The version and release has the format *z/OS VxxRyy* where *V* stands for version, *xx* is the version number, *R* stands for release, and *yy* is the release number. For example, *z/OS V2R1*.

JES-member-name
JES2 multi-access spool (MAS) member name or JES3 complex member name that is assigned to the primary job entry subsystem (JES) that is running on the system.

JES-type
Type for the primary job entry subsystem running on the system. The type is either JES2 or JES3.

CPC-name
Name specified for the central processor complex (CPC) at the support element (SE) of that processor complex.

CPC-serial
Serial number of the CPC.

proxy-name
Name of the HTTP proxy definition that specifies the settings required to access the system through an HTTP or SOCKS proxy server.

server-name
Name of the server definition that specifies the settings required to access the FTP or SFTP server that is running on the system.

total-items
Number of system definitions that were retrieved.

Example

In the following example, the GET method is used to retrieve a list of the systems that are defined to the z/OSMF instance with host name *zosmf1.yourco.com* and that are included in CPC *CPC1*.

```
GET /zosmf/resttopology/systems/cpcName/CPC1 HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 177. Sample request to retrieve a list of systems included in a CPC

A sample response is shown in Figure 178.

```
HTTP/1.1 200 OK
Date: Thu, 15 Feb 2015 05:39:28 +0000GMT
Connection: close

{
 "items":[
   {
    "systemNickName":"sys1",
    "systemName":"sys1",
    "sysplexName":"plex1",
    "groupNames":"test,development",
    "url":"https://zosmf1.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY1",
    "jesType":"JES2",
    "cpcName":"CPC1",
    "cpcSerial":"30104",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   },
   {
    "systemNickName":"sys10",
    "systemName":"sys10",
    "sysplexName":"plex2",
    "groupNames":"production",
    "url":"https://zosmf10.yourco.com/zosmf/",
    "zosVR":"z/OS V2R1",
    "jesMemberName":"SY10",
    "jesType":"JES2",
    "cpcName":"CPC1",
    "cpcSerial":"30104",
    "httpProxyName":"No Proxy",
    "ftpDestinationName":"IBM-testcase-mvs"
   }
 ],
 "numRows":"2"
}
```

Figure 178. Sample response from a request to retrieve a list of systems included in a CPC

TSO/E address space services

The TSO/E address space services is an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. A set of REST services is provided for working with TSO/E address spaces on a z/OS system, as described in this topic.

Table 195 lists the operations that the TSO/E address space services provide.

Table 195. Operations provided through the TSO/E address space services.

Operation	HTTP method and URI path
"Start or reconnect to a TSO/E address space" on page 334	POST /zosmf/tsoApp/tso?<*parms*>
"Start an application in a TSO/E address space" on page 337	POST /zosmf/tsoApp/app/<*servletKey*>/<*appKey*>
"Receive messages from a TSO/E address space" on page 345	GET /zosmf/tsoApp/tso/<*servletKey*>
"Receive messages from an application" on page 347	GET /zosmf/tsoApp/app/<*servletKey*>/<*appKey*>
"Send messages to a TSO/E address space" on page 339	PUT /zosmf/tsoApp/tso/<*servletKey*>?[readReply=true | false]
"Send messages to an application" on page 341	PUT /zosmf/tsoApp/app/<*servletKey*>/<*appKey*>
"Ping a TSO/E address space" on page 343	PUT /zosmf/tsoApp/tso/ping/<*servletKey*>
"End a TSO/E address space" on page 349	DELETE /zosmf/tsoApp/tso/<*servletKey*>?[tsoforcecancel=true | false]

Required authorizations

Generally, your z/OSMF user ID requires the same authorizations for using the TSO/E address space services as when you perform these operations through a TSO/E session on the z/OS system. For example, to start an application in a TSO/E address space requires that your user ID be authorized to operate that application.

In addition, you must be logged in to z/OSMF, and your user ID must have READ access to SAF resource `CEA.CEATSO.*` in the SERVAUTH class. If your installation is using RACF and you want to assign administrators and users READ access to the resource, you can create a profile like the following:

```
//* Permit the Administrators group to this profile            *
 PERMIT CEA.CEATSO.* CLASS(SERVAUTH) ID(IZUADMIN) ACCESS(READ)

//* Permit the Users group to this profile                     *
 PERMIT CEA.CEATSO.* CLASS(SERVAUTH) ID(IZUUSER) ACCESS(READ)

//* Permit the started task USERID to this profile             *
 PERMIT CEA.CEATSO.* CLASS(SERVAUTH) ID(IZUSVR) ACCESS(READ)

//* Make changes effective                                     *
 SETROPTS RACLIST(SERVAUTH) REFRESH
```

For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Content type used for HTTP request and response data

The JSON content type (`"Content-Type: application/json"`) is used for request and response data. The following JSON object is used by all TSO/E address space services for returning data and status about the requested operations. The attributes provided in the JSON object depend on the requested operation.

```
{
 "sevletKey": "serlet-key",
 "ver": "structure-version",
 "queueID": "message queue ID",
 "tsoData": "TSO/E-messages",
 "appData": "application-messages",
 "timeout": "timeout-indicator",
 "reused": "reconnected-indicator",
 "msgData": "z/OSMF-messages"
 "messages": "unexpected z/OSMF-messages"
}
```

where:

servlet-key
: Unique identifier for the servlet entry. It maps to the TSO/E address space ID and provides additional information about the address space. To communicate with the TSO/E address space, the client must provide the servlet key.

structure-version
: Version of the TSO/E address space services and the JSON object structure used for this request. The version sequence starts at 0100, and is incremented only if the services or the JSON structure changes. In your application, check the value of the returned structure and verify that your application is compatible with the current API.

message queue ID
: When the TSO/E address space interface starts a new TSO/E session, it also creates a new z/OS UNIX message queue to enable communication between the client and the TSO/E address space. This value is the identifier for the z/OS UNIX message queue.

TSO/E-messages
: TSO/E messages that were received during the request. The *tsoData* attribute is included in the JSON object only if TSO/E messages were received.

  The value returned in the *tsoData* attribute is a JSON object that describes the messages that were received. For example, the TSO/E message JSON format has the following syntax:

  ```
  {"message-type":{"VERSION":"JSON-version","data-type":"data-value"}}
  ```

  where:

  message-type
  : Keyword that identifies the type of TSO/E message. The value can be TSO MESSAGE, TSO PROMPT, or TSO RESPONSE.

  JSON-version
  : A four-digit number that identifies the JSON version used to format the message.

  data-type
  : Keyword that describes the type of data included in the *data-value* variable. The value can be DATA, HIDDEN, or ACTION.

  Example: {"TSO RESPONSE":{"VERSION":"0100","DATA":"ALLOC DA"}}

application-messages
: Messages received during the request from an application running in a TSO/E address space. The

appData attribute is included in the JSON object only if messages were received from an application and no TSO/E messages were received during the request.

timeout-indicator
Indicator of whether the request timed out while waiting for a response. The value is *true* if the request timed out. Otherwise, the value is *false*.

If the service creates a new TSO/E address space, the service attempts to read the initial startup TSO/E messages. If no messages are received in the time allotted, this value is set to *true*.

If the service reconnects the user to an existing TSO/E address space, no startup messages are expected; therefore, the service does not wait for any startup TSO/E messages.

reconnected-indicator
Indicator of whether the service connected the user to an existing TSO/E session instead of a new session. The *reused* attribute is included in the JSON object only if the *appsessid* parameter is provided for the start TSO/E address space request. The value returned for the *reused* attribute is *true* if a TSO/E address space with that *appsessid* exists. Otherwise, the value is *false*.

z/OSMF-messages
z/OSMF messages received during the request. The *messages* attribute is included in the JSON object only if an error occurred during the request. The message ID, message text, and stack trace are provided for each z/OSMF message received.

unexpected z/OSMF messages
z/OSMF messages received for unexpected errors.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains a message that describes the error.

The following HTTP status codes are valid:

HTTP 200 OK
Success.

HTTP 400 Bad request
Request contained incorrect parameters.

HTTP 401 Unauthorized
Submitter of the request did not authenticate to z/OSMF or is not authorized to use the TSO/E address space services.

HTTP 404 Bad URL
Target of the request (a URL) was not found.

HTTP 500 Internal server error
Programming error.

Error logging

Errors from the TSO/E address space services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Start or reconnect to a TSO/E address space

You can use this operation to start a new TSO/E address space or to reconnect to a dormant TSO/E address space.

HTTP method and URI path

```
POST /zosmf/tsoApp/tso?<parms>
```

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **tso** informs the service that the request is for a TSO/E address space.
- **<parms>** qualifies the request with one or more of the parameters described in Table 196.

When the TSO/E address space interface starts a new TSO/E session, it also creates a new z/OS UNIX message queue to enable communication between the client and the TSO/E address space.

When the interface reconnects to a dormant TSO/E address space, the interface reuses the session resources including the z/OS UNIX message queue.

Note: A *dormant TSO/E address space* is an address space that has been deactivated for communication through its z/OS UNIX message queue but remains available at a TSO/E READY prompt for a period of time.

Supported parameters

Table 196. Supported parameters for the start and reconnect TSO/E session requests

Parameter	Required	Description
proc	Yes	Name of the TSO/E logon procedure to use to log onto the TSO/E address space.
chset	Yes	Character set to use for the caller's TSO/E address space. This value is used by the applications running in the TSO/E address space to convert messages and responses from UTF-8 to EBCDIC. The default character set, which is 697 decimal, will be used if zero is specified as the value.
cpage	Yes	Codepage to use for the caller's TSO/E address space. This value is used by the applications running in the TSO/E address space to convert messages and responses from UTF-8 to EBCDIC. The default codepage, which is 1047 decimal, will be used if zero is specified as the value.
rows	Yes	Number of rows to be displayed on the screen. The default number of rows, which is 24, will be used if zero is specified as the value.
cols	Yes	Number of columns to be displayed on the screen. The default number of columns, which is 80, will be used if zero is specified as the value.
acct	No	TSO/E user account number.
ugrp	No	Name of the TSO/E user group.
rsize	No	Region size to use for the TSO/E address space.

Table 196. Supported parameters for the start and reconnect TSO/E session requests (continued)

Parameter	Required	Description
appsessid	No (for new), Yes (for reconnect)	Identifier that uniquely identifies the TSO/E address space. This parameter is optional when starting a new TSO/E address space, and it is required when reconnecting to an existing TSO/E address space. If an address space with the specified identifier does not exist, a new TSO/E address space will be created and assigned the identifier specified.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 331.

In addition, only the z/OSMF user that started the TSO/E address space is authorized to use the z/OS UNIX message queue associated with that address space.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes a JSON object with additional information about the results of the request. For more details, see "Content type used for HTTP request and response data" on page 332.

Example: Creating a new TSO/E address space

To create a new address space with the following settings, submit the request depicted in Figure 179 on page 336:

- Procedure name: IKJACCNT
- Character set: 697
- Code page: 1047
- Screen rows: 204
- Screen columns: 160
- Region size: 50000

- Account number: DEFAULT

```
POST /zosmf/tsoApp/tso?proc=IKJACCNT&chset=697&cpage=1047&rows=204
&cols=160&rsize=50000&acct=DEFAULT HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 179. Sample request to create a new TSO/E address space

A sample response is shown in Figure 180.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","queueID":"4","ver":"0100","tsoData":[{"TSO MESSAGE":
{"VERSION":"0100","DATA":"ZOSMFAD LOGON IN PROGRESS AT 11:27:28 ON
OCTOBER 12, 2011"}},{"TSO MESSAGE":{"VERSION":"0100","DATA":"NO BROADCAST
MESSAGES"}}],"timeout":false, "reused":false}
```

Figure 180. Sample response from create TSO/E address space request

Example: Reconnecting to an existing TSO/E address space

To reconnect to the TSO/E address space associated with application session ID *sdsf_23715376543765*, specify the following settings, and submit the request depicted in Figure 181:

- Procedure name: IKJACCNT
- Character set: 697
- Code page: 1047
- Screen rows: 204
- Screen columns: 160
- Region size: 50000
- Account number: DEFAULT
- Application Session ID: sdsf_23715376543765

```
POST /zosmf/tsoApp/tso?proc=IKJACCNT&chset=697&cpage=1047&rows=204
&cols=160&rsize=50000&acct=DEFAULT&appsessid=sdsf_23715376543765 HTTP/1.1

Host: zosmf1.yourco.com
```

Figure 181. Sample request to reconnect to an existing TSO/E address space

A sample response is shown in Figure 182.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","queueID":"4","ver":"0100","timeout":false,"reused":true}
```

Figure 182. Sample response from a reconnect to TSO/E address space request

Start an application in a TSO/E address space

You can use this operation to start an application in a TSO/E address space.

HTTP method and URI path

```
POST /zosmf/tsoApp/app/<servletKey>/<appKey>
```

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **app** informs the service that the request is for an application running in a TSO/E address space.
- **<servletKey>** identifies the TSO/E address space in which to start the application.
- **<appKey>** identifies the application to be started.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that contains the command required to start the application, for example:

```
{"startcmd": "{command} {INMSG} 0x4 {OUTMSG} 0x8004"}
```

Figure 183. Starting an application: request content

where:

{command}
Any TSO/E command that can be used to start the application. For example, the `ISFWEB` parameter starts the SDSF application.

{INMSG} 0x4 {OUTMSG} 0x8004
The API uses a z/OS UNIX message queue to facilitate communication between the client and the application running in a TSO/E address space. Specifically, the API assigns an input and output message type ID to be used when reading messages from the application (output messages) or writing messages to the application (input messages).

The input message type ID is `0x4` (hexadecimal 4) and output message type ID is `0x8004` (hexadecimal 8004). You must specify variables that the application will recognize. For example, if the application recognizes ISFMSGTYPEW as the input message type and ISFMSGTYPER as the output message type, specify the following value:

```
ISFMSGTYPEW 0x4 ISFMSGTYPER 0x8004
```

Processing overview

When the client requests to start an application in a TSO/E address space, the API completes the following actions:

- Assigns input and output message types to use for communication with the application to be started. The message types will be used only for the application identified by the **appKey**.
- Replaces the variables in the command with the assigned message types.
- Sends the TSO/E command to the TSO/E address space identified by the servlet key.
- Attempts to read a TSO/E or application response message. If no messages are received in the time allotted, a timeout indication will be returned. Any TSO/E messages are prioritized over application messages. Typically, when a caller receives a TSO/E message while attempting to receive application messages, the caller processes the TSO/E messages, then attempts to retrieve the queued application messages.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes a JSON object that contains the application response messages, or a timeout indication. For more details, see "Content type used for HTTP request and response data" on page 332.

Example

To start application *BkApp001* in the TSO/E address space associated with servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 184.

```
POST /zosmf/tsoApp/app/ZOSMFAD-71-aabcaaaf/BkApp001 HTTP/1.1
Host: zosmf1.yourco.com

{"startcmd":"ISFWEB ISFMSGTYPEW 0x4 ISFMSGTYPER 0x8004"}
```

Figure 184. Sample request to start an application in a TSO/E address space

A sample response is shown in Figure 185.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100",
"appData":...,"timeout":false}
```

Figure 185. Sample response from a start an application in a TSO/E address space request

Send messages to a TSO/E address space

You can use this operation to send TSO/E messages to a TSO/E address space.

HTTP method and URI path

```
PUT /zosmf/tsoApp/tso/<servletKey>?[readReply=true|false]
```

---

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **tso** informs the service that the request is for a TSO/E address space.
- **<servletKey>** identifies the TSO/E address space to which the message will be sent.
- **[readReply]** is an optional parameter that indicates whether the service should send the message and immediately check for a response (default) or just send the message. To immediately check for a response, omit the parameter or set its value to *true*. Otherwise, set its value to *false*.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that describes the message to be sent. For example, the TSO/E message JSON format has the following syntax:

```
{"message-type":{"VERSION":"JSON-version","data-type":"data-value"}}
```

where:

message-type
: Keyword that identifies the type of TSO/E message. The value can be TSO MESSAGE, TSO PROMPT, or TSO RESPONSE.

JSON-version
: A four-digit number that identifies the JSON version used to format the message.

data-type
: Keyword that describes the type of data included in the *data-value* variable. The value can be DATA, HIDDEN, or ACTION.

Example: {"TSO RESPONSE":{"VERSION":"0100","DATA":"ALLOC DA"}}

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, note that the API will attempt to read response TSO/E messages after the input message is sent. If no TSO/E messages are received after a predetermined time period, a timeout indication will be returned.

Required authorizations

See “Required authorizations” on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 333.

The response also includes a JSON object that contains the TSO/E response messages, or a timeout indication. For more details, see “Content type used for HTTP request and response data” on page 332.

Example

To send a TSO/E message to the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf* and read the response TSO/E messages, submit the request depicted in Figure 186.

```
PUT /zosmf/tsoApp/tso/ZOSMFAD-71-aabcaaaf HTTP/1.1
Host: zosmf1.yourco.com

{"TSO RESPONSE":{"VERSION":"0100","DATA":"TIME"}}
```

Figure 186. Sample request to send a message to a TSO/E address space

A sample response is shown in Figure 187.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","tsoData":[{"TSO MESSAGE":
{"VERSION":"0100","DATA":"TIME-12:09:07 PM. CPU-00:00:00 SERVICE-92319
SESSION-00:00:13 OCTOBER 12,2011"}}],"timeout":false}
```

Figure 187. Sample response from send message to TSO/E address space request

Send messages to an application

You can use this operation to send messages to an application running in a TSO/E address space.

HTTP method and URI path

```
PUT /zosmf/tsoApp/app/<servletKey>/<appKey>
```

---

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **app** informs the service that the request is for an application running in a TSO/E address space.
- **<servletKey>** identifies the TSO/E address space where the application is running.
- **<appKey>** identifies the application to which to send messages.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

Your request must include a JSON object that contains the application message to be sent.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, note that the API will attempt to read application and TSO/E response messages after the application input message is sent. If no messages are received in the time allotted, a timeout indication will be returned.

Any TSO/E messages are prioritized over application messages. Typically, when a caller receives a TSO/E message while attempting to receive application messages, the caller processes the TSO/E messages, then attempts to retrieve the queued application messages.

Required authorizations

See "Required authorizations" on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes a JSON object that contains the application response messages, or a timeout indication. For more details, see "Content type used for HTTP request and response data" on page 332.

Example

To send a message to application *BkApp001*, which is running in the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 188.

```
PUT /zosmf/tsoApp/app/ZOSMFAD-71-aabcaaaf/BkApp001 HTTP/1.1
Host: zosmf1.yourco.com

{...}
```

Figure 188. Sample request to send a message to an application

A sample response is shown in Figure 189.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","appData":[...],"timeout":false}
```

Figure 189. Sample response from send message to an application request

Ping a TSO/E address space

You can use this operation to ping a TSO/E address space. Doing so at regular intervals helps to ensure that the TSO/E address space remains active for the client. Otherwise, the server can end the TSO/E address space without warning.

HTTP method and URI path

```
PUT /zosmf/tsoApp/tso/ping/<servletKey>
```

---

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **tso** informs the service that the request is for a TSO/E address space.
- **ping** informs the service to ping the specified TSO/E address space.
- **<servletKey>** identifies the TSO/E address space for the service to ping.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, note that each TSO/E address space has an idle application time that the TSO/E address space services interface uses to determine if the client application associated with the address space is active. If the idle application time is 10 minutes, the client application is considered to be inactive. In which case, the API ends all the TSO/E address spaces associated with the client application.

To prevent TSO/E address spaces from ending because of idle application time, callers can issue a ping request at least once every 5 minutes. Doing so informs the TSO/E address space services interface that the client application is still active, and causes the interface to reset the idle application time for all the TSO/E address spaces associated with the client application.

Required authorizations

See “Required authorizations” on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 333.

The response also includes a JSON object that contains all the attributes in the JSON structure except the message data. For more details, see “Content type used for HTTP request and response data” on page 332.

Example

To ping the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 190.

```
PUT /zosmf/tsoApp/ping/ZOSMFAD-71-aabcaaaf HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 190. Sample request to ping a TSO/E address space

A sample response is shown in Figure 191.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","timeout":false}
```

Figure 191. Sample response from ping TSO/E address space request

Receive messages from a TSO/E address space

You can use this operation to receive messages from a TSO/E address space.

HTTP method and URI path

```
GET /zosmf/tsoApp/tso/<servletKey>
```

---

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **tso** informs the service that the request is for a TSO/E address space.
- **<servletKey>** identifies the TSO/E address space from which to receive messages.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, note that the API will attempt to read TSO/E messages. If no TSO/E messages are received after 15 seconds, a timeout indication will be returned.

Required authorizations

See "Required authorizations" on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 333.

The response also includes a JSON object that contains the TSO/E response messages, or a timeout indication. For more details, see "Content type used for HTTP request and response data" on page 332.

Example

To read TSO/E messages from the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 192 on page 346.

```
GET /zosmf/tsoApp/tso/ZOSMFAD-71-aabcaaaf HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 192. Sample request to receive a message from a TSO/E address space

A sample response is shown in Figure 193.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","tsoData":
[{"TSO MESSAGE":{"VERSION":"0100","DATA":"--> LOGON proc version = 04/28/2011"}},
{"TSO MESSAGE":{"VERSION":"0100","DATA":"  "}},
{"TSO MESSAGE":{"VERSION":"0100","DATA":"--> System Name   = DCEIMGNE"}},
{"TSO MESSAGE":{"VERSION":"0100","DATA":"--> System Suffix = NE"}},
{"TSO MESSAGE":{"VERSION":"0100","DATA":"--> SYSPLEX Name  = CFCIMGNE"}},
{"TSO MESSAGE":{"VERSION":"0100","DATA":"--> SYSRES Volume = SD1131"}},
{"TSO MESSAGE":{"VERSION":"0100","DATA":"  "}}],
"timeout":false}
```

Figure 193. Sample response from receive message from a TSO/E address space request

Receive messages from an application

You can use this operation to receive messages from an application running in a TSO/E address space.

HTTP method and URI path

```
GET /zosmf/tsoApp/app/<servletKey>/<appKey>
```

---

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **app** informs the service that the request is for an application running in a TSO/E address space.
- **<servletKey>** identifies the TSO/E address space where the application is running.
- **<appKey>** identifies the application to which to send messages.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, note that the API will attempt to read application and TSO/E response messages. If no messages are received in the time allotted, a timeout indication will be returned.

Any TSO/E messages are prioritized over application messages. Typically, when a caller receives a TSO/E message while attempting to receive application messages, the caller processes the TSO/E messages, then attempts to retrieve the queued application messages.

Required authorizations

See “Required authorizations” on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 333.

The response also includes a JSON object that contains the application response messages, or a timeout indication. For more details, see “Content type used for HTTP request and response data” on page 332.

Example

To receive TSO/E or application messages from application *BkApp001*, which is running in the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 194.

```
GET /zosmf/tsoApp/app/ZOSMFAD-71-aabcaaaf/BkApp001 HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 194. Sample request to receive messages from an application

A sample response is shown in Figure 195.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","appData":[...],"timeout":false}
```

Figure 195. Sample response for request to receive messages from an application

End a TSO/E address space

You can use this operation to end a TSO/E address space or place it in a dormant state as a candidate for reconnection.

HTTP method and URI path

```
DELETE /zosmf/tsoApp/tso/<servletKey>?[tsoforcecancel=true|false]
```

---

where:

- **zosmf/tsoApp** identifies the TSO/E address space services.
- **tso** informs the service that the request is for a TSO/E address space.
- **<servletKey>** identifies the TSO/E address space to be ended or placed in a dormant state.
- **[tsoforcecancel]** is an optional parameter that indicates whether to use the CANCEL or LOGOFF command to end the TSO/E address space. The parameter can have one of the following values:
  - **True**: The CANCEL command will be issued and the TSO/E address space will not be placed in a dormant state.
  - **False** (default): The LOGOFF command will be issued. If the CEA reconnect feature is enabled in your installation, the TSO/E address space will be placed in a dormant state. Otherwise, the TSO/E session will end.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Required authorizations

See “Required authorizations” on page 331.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see “Error handling” on page 333.

The response also includes a JSON object with additional information about the results of the request. For more details, see “Content type used for HTTP request and response data” on page 332.

Example: Logging off a TSO/E address space

To use the LOGOFF command to end the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 196.

```
DELETE /zosmf/tsoApp/tso/ZOSMFAD-71-aabcaaaf HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 196. Sample request to logoff a TSO/E address space

A sample response is shown in Figure 197.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","timeout":false,"reuse":false}
```

Figure 197. Sample response for logoff a TSO/E address space request

Example: Canceling a TSO/E address space

To use the CANCEL command to end the TSO/E address space identified by servlet key *ZOSMFAD-71-aabcaaaf*, submit the request depicted in Figure 198.

```
DELETE /zosmf/tsoApp/tso/ZOSMFAD-71-aabcaaaf?tsoforcecancel=true HTTP/1.1
Host: zosmf1.yourco.com
```

Figure 198. Sample request to cancel a TSO/E address space

A sample response is shown in Figure 199.

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2011 05:39:28 +0000GMT
Connection: close

{"servletKey":"ZOSMFAD-71-aabcaaaf","ver":"0100","timeout":false,"reuse":true}
```

Figure 199. Sample response for a cancel TSO/E address space request

WLM resource pooling services

The WLM resource pooling services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. The WLM resource pooling services provide a programming interface for WLM policy elements. You can work with WLM policy elements in the context of IBM Cloud Provisioning and Management for z/OS.

With the WLM resource pooling services, you can provision and deprovision WLM policy elements, dynamically construct a new service definition, and install the service definition.

Table 197 lists the operations that the WLM resource pooling services provide.

Table 197. Operations provided through the WLM resource pooling services.

Operation	HTTP method and URI path
"Prime a WLM resource pool" on page 353	`POST /zosmf/zwlm/rest/wrps`
"Delete a WLM resource pool" on page 356	`DELETE /zosmf/zwlm/rest/wrps/`*`wrpid`*
"Construct a WLM service definition" on page 358	`PUT /zosmf/zwlm/rest/policy/inspolicy`
"Construct a WLM service definition with remove and install" on page 360	`PUT /zosmf/zwlm/rest/policy/inspolicy`

Required authorizations

The user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: <SAF-prefix>.ZOSMF.RESOURCE_POOL.WLM.*domainid*, where *domainid* is the identifier of the domain of systems.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a `4nn` code or a `5nn` code. The HTTP status codes are described in the topics for the individual services.

In addition, some errors might also include a returned JSON object that contains a message that describes the error. You can use this information to diagnose the error or provide it to IBM Support, if required.

The following HTTP status codes are valid:

HTTP 200 OK
: Request was processed successfully.

HTTP 204 No Content
: Request was processed successfully.

HTTP 400 Bad request
: Request could not be processed because it contains a syntax error or an incorrect parameter.

HTTP 401 Unauthorized
: Request could not be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both, or the client did not authenticate to z/OSMF.

HTTP 500 Internal server error
: Server encountered an error. See the response body for a JSON object with information about the error.

Error logging

Errors from the WLM resource pooling services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Prime a WLM resource pool

Use this operation to create a record for a WLM resource pool.

HTTP method and URI path

```
POST /zosmf/zwlm/rest/wrps
```

Query parameters

None.

Description

This operation creates a WRP record in a persistent file for WRP data. In addition, it causes a new WRP record to be displayed on the WLM Resource Pool page of the z/OSMF Workload Management task. The actual provisioning for the report class, and the definition and installation of the service definition, occurs only when the WRP definition is completed by the WLM administrator in z/OSMF.

On successful completion of a prime a WLM resource pool request, a response body that describes the request is returned.

For the properties that you can specify on the request body, see "Request content."

For a description of the response content, see "Response content" on page 354.

Request content

The request content is expected to contain a JSON object. See Table 198 for a description of the fields.

Table 198. Request content for the prime WLM resource pool request

Field name	Required or Optional	Description
cloud-info	Required	Specifies the attributes of the cloud: **domain-name** Name of the domain of systems **domain-id** Generated identifier of the domain of systems **tenant-name** Name of the tenant for the domain **tenant-id** Generated identifier of the tenant **template-name** Name of the software services template

Table 198. Request content for the prime WLM resource pool request (continued)

Field name	Required or Optional	Description
wrp-data	Required	Specifies the attributes of the WLM resource pool: **wrp-name** Name of the WLM resource pool **service-level-agreements** Array of service-level agreements, in the form "sla-name":"level", for example: `"service-level-agreements": [ {"SlaOne":"GOLD"}, {"SlaTwo":"SILVER"}, ]` **report-class-name** Name of the report class

Authorization requirements

See "Required authorizations" on page 351.

HTTP status codes

For a successful request, the response body is provided, as described in "Request content" on page 353.

For a list of status codes, see "Error handling" on page 351.

Response content

On successful completion, the service returns a response body, which contains a JSON object. For a description of fields in the JSON object, see Table 199.

Table 199. Response content for a successful prime WLM resource pool request

Field name	Description
status	Status of the request.
return-code	Return code of the request.
message	Message issued for the request.
wrp-id	Identifier of the WLM resource pool
state	State of the request.

Example HTTP interaction

1. The example in Figure 200 on page 355 shows a request to prime a WLM resource pool.

```
POST  https://host:port/zosmf/zwlm/rest/wrps

{
  "cloud-info":{
                "domain-name":"DOMAIN1",
                "domain-id":"12345ABC",
                "tenant-name":"Joey",
                "tenant-id":"IZU$000,
                "template-name":"CICSBasic"
               }

   "wrp-data"{
                "wrp-name":"WRP1",
                "service-level-agreements":[
                                           {"sla-name":"GOLD"},
                                           {"sla-name":"SILVER"}
                                           ],
                "report-class-name":"Joey00"
       }
}
```

Figure 200. Sample request to issue a prime WRP request

The following is the response body for the request.

```
{
  "status":"success',
  "return-code":"0",
  "message":null,
  "wrp-id":"1090f34e-0a5a-4506-b553-91e932a46f3e",
  "wrp-name":"WRP1",
  "state":"initialized"
}
```

Figure 201. Sample response body

Delete a WLM resource pool

Use this operation to delete the record for a WLM resource pool.

HTTP method and URI path

```
DELETE /zosmf/zwlm/rest/wrps/wrpid
```

In this request, *wrpid* is the identifier of the WLM resource pool.

Query parameters

None.

Description

This operation:

- Removes a WLM resource pool record from the persistent file for WLM resource pools
- Deprovisions the report class in the current installed service definition if it is not referenced by any other classification rule
- Deletes the WLM resource pool record from the WLM Resource Pool page of the z/OSMF Workload Management task.

On successful completion, a response body that describes the request is returned.

For a description of the response content, see "Response content."

Request content

None.

Authorization requirements

See "Required authorizations" on page 351.

HTTP status codes

For a successful request, the response body is provided, as described in "Request content."

For a list of status codes, see "Error handling" on page 351.

Response content

On completion, the service returns a response body, which contains a JSON object. For a description of fields in the JSON object, see Table 200.

Table 200. Response content for a delete WLM resource pool request

Field name	Description
status	Status of the request.
return-code	Return code of the request.
message	Message issued for the request.

Example HTTP interaction

1. The example in Figure 202 shows a request to delete a WLM resource pool.

```
DELETE  https://host:port/zosmf/zwlm/rest/wrps/1090f34e-0a5a-4506-b553-91e932a46f3e
```

Figure 202. Sample request to issue a delete WLM resource pool request

The following is the response body for the successful request.

```
{
  "status":"success',
  "return-code":"0",
  "message":null
}
```

Figure 203. Sample response body

Construct a WLM service definition

Use this operation to construct a new service definition based on the current installed service definition.

HTTP method and URI path

```
PUT /zosmf/zwlm/rest/policy/inspolicy
```

Query parameters

None.

Description

This operation constructs a new service definition based on the current installed service definition. A description for the report class is generated based on the domain name and domain ID.

On successful completion, a response body that describes the request is returned.

For the properties that you can specify on the request body, see "Request content."

For a description of the response content, see "Response content" on page 359.

Request content

The request content is expected to contain a JSON object. See Table 201 for a description of the fields.

Table 201. Request content for the construct a WLM service definition request

Field name	Required or Optional	Description
cloud-info	Required	Specifies attributes of the cloud: **WRP-ID** Identifier of the WLM resource pool.

provision-data	Required	Specifies the attributes of the WLM service definition:
		classification-rules
		Array of attributes, in the form "attribute-name":"value", for example:
		`"classification-rules":`
		`[`
		`{"service-level-agreement":"Gold",`
		`"qualifier-value":"CICSL00",`
		`"report-class": "optional"}`
		`]`

Authorization requirements

See "Required authorizations" on page 351.

HTTP status codes

For a successful request, the response body is provided, as described in "Request content" on page 358.

For a list of status codes, see "Error handling" on page 351.

Response content

On successful completion, the service returns a response body, which contains a JSON object. For a description of fields in the JSON object, see Table 202.

Table 202. Response content for a successful construct a WLM service definition request

Field name	Description
status	Status of the request.
messages	Message issued for the request.

result	Result for the request.
	
	classification-rules
	Array of attributes, in the form "cl-rule-id":"value", for example:
	`"classification-rules":`
	`[`
	`{"cl-rule-id":"`*id*`"}`
	`]`

Example HTTP interaction

1. The example in Figure 204 shows a request to construct a service definition based on the current installed definition.

```
PUT  https://host:port/zosmf/zwlm/rest/policy/inspolicy

{
    "cloud-info":{
     "WRP-ID":"e5697dd6-88da-43f8-8f89-bbdf9537b296",
                }
    "provision-data":{

                       classification-rules: [
                                              {"service-level-agreement":"Gold",
                                               "qualifier-value":"CICSL00",
                                               "report-class": "optional"}
                                             ]
                      }
}
```

Figure 204. Sample request to construct a service definition based on the current installed definition

The following is the response body for the request.

```
{
   "state":"success",
    "return-code":"0",
    "message":null,
    "result":{
              "classification-rules":[
                                       { "cl-rule-id":"1090f34e-0a5a-4506-b553-91e932a46f3e"}
                                     ]
             }
}
```

Figure 205. Sample response body

Construct a WLM service definition with remove and install

Use this operation to construct a new service definition by removing the classification rule, then installing the new service definition.

HTTP method and URI path

```
PUT /zosmf/zwlm/rest/policy/inspolicy
```

Query parameters

None.

Description

This operation constructs a new service definition by removing the classification rule, then installing the new service definition.

On successful completion, a response body that describes the request is returned.

For the properties that you can specify on the request body, see "Request content."

For a description of the response content, see "Response content" on page 361.

Request content

The request content is expected to contain a JSON object. See Table 203 for a description of the fields.

Table 203. Request content for the construct a WLM service definition request

Field name	Required or Optional	Description
cloud-info	Required	Specifies cloud-related attributes: **WRP-ID** Identifier of the WLM resource pool

deprovision-data	Required	Specifies the attributes of the WLM service definition:
		classification-rules
		Array of attributes, in the form "attribute-name":"value", for example:
		`"classification-rules":`
		`[`
		`{"cl-rule-id":"`*value*`"}`
		`]`

Authorization requirements

See "Required authorizations" on page 351.

HTTP status codes

For a successful request, the response body is provided, as described in "Request content" on page 360.

For a list of status codes, see "Error handling" on page 351.

Response content

On successful completion, the service returns a response body, which contains a JSON object. For a description of fields in the JSON object, see Table 204.

Table 204. Response content for a successful construct a WLM service definition request

Field name	Description
state	State of the request.
messages	Message issued for the request.

Example HTTP interactions

1. The example in Figure 206 shows a request to construct a service definition by removing the classification rule, then installing the new service definition.

```
PUT  https://host:port/zosmf/zwlm/rest/policy/inspolicy

{   "cloud-info": {
       "WRP-ID":"e5697dd6-88da-43f8-8f89-bbdf9537b296",
                }
    "deprovision-data":{
                      "classification-rules":[
                                             { "cl-rule-id":"1090f34e-0a5a-4506-b553-91e932a46f3e"}
                                             ]
                      }
}
```

Figure 206. Sample request to construct a service definition by removing the classification rule, then installing the new service definition

The following is the response body for the request.

```
{
   "state":"success",
   "message":null,
}
```

Figure 207. Sample response body

z/OS console services

The z/OS console services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. The z/OS console services provide a programming interface for performing z/OS console operations.

With the z/OS console services, you can issue system commands and work with both solicited messages (messages that were issued in response to the command) and unsolicited messages (other messages that might or might not have been issued in response to the command). z/OS console services establish an extended MCS (EMCS) console, which is then used to issue commands and receive messages.

Table 205 lists the operations that the z/OS console services provide.

Table 205. Operations provided through the z/OS console services.

Operation	HTTP method and URI path
"Issue a command" on page 364	`PUT /zosmf/restconsoles/consoles/`*`consolename`* `PUT /zosmf/restconsoles/consoles/defcn`
"Get a command response" on page 372	`GET /zosmf/restconsoles/consoles/`*`console-name`*`/solmsgs/C`*`key-number`* `GET /zosmf/restconsoles/consoles/defcn/solmsgs/C`*`key-number`*
"Get the detect result for unsolicited messages" on page 376	`GET /zosmf/restconsoles/consoles/`*`consolename`*`/detections/D`*`key-number`* `GET /zosmf/restconsoles/consoles/defcn/detections/D`*`key-number`*

Required authorizations

Your user ID must have the same authority when issuing a command with the z/OS console services as when issuing a command through a console on a z/OS system.

The required authority is:

- READ access to the MVS.MCSOPER.*consolename* resource in the OPERCMDS class, where *consolename* is the name of the EMCS console that is used to issue the command
- READ access to the CONSOLE resource in the TSOAUTH class.

z/OS console services use z/OSMF TSO/E address space services to create a TSO address space as the host for an EMCS console. To use TSO/E address space services, you must have:

- READ access to resource *account* in class ACCTNUM, where *account* is the value specified in the COMMON_TSO ACCT option in parmlib
- READ access to resource CEA.CEATSO.TSOREQUEST in class SERVAUTH
- READ access to resource *proc* in class TSOPROC, where *proc* is the value specified with the COMMON_TSO PROC option in parmlib.

You must also ensure that the z/OSMF started task user ID, which is IZUSVR by default, has READ access to resource CEA.CEATSO.TSOREQUEST in class SERVAUTH.

The TSO/E address space services authority might already be defined if you are using z/OS data set and file REST services, as those services require similar authority.

Configuration

z/OS console services use the TSO CONSOLE command to establish an EMCS console, which allows you to issue system commands and retrieve the messages that are issued in response. Console attributes such as ROUTCODE and AUTH affect the messages that the EMCS console can receive and the commands that the console can issue. When you use the z/OS console services, be sure that the EMCS console that is established has the desired attributes. For information, see EMCS consoles in *z/OS MVS Planning: Operations*.

In addition, be aware that messages can be suppressed due to settings in the active MPFLST*xx* member of parmlib. If a message associated with a command response is suppressed, a REST API call that attempts to detect that message will fail.

You can use SAF to control console attributes. The RACF ADDUSER command with the OPERPARM parameter sets console attributes when a user establishes an EMCS console. Using the ADDUSER command to control console attributes requires that you know in advance the name of the EMCS console that the z/OS Console service will use. The console name will be either a name that you specify on the Issue Command service, or a name that the service generates, as described in "Issue a command" on page 364.

For example, if user CJOEY plans to accept the default console name, CJOEYCN, he could issue this RACF command to set console attributes for the console:

```
ADDUSER CJOEYCN OPERPARM(AUTH(MASTER) ROUTCODE(ALL))
```

To control the parameters that z/OS console services use when creating a TSO address space as the host for an EMCS console, use parmlib option COMMON_TSO ACCT(IZUACCT) REGION(50000) PROC(IZUFPROC). Configure this setting before z/OS console services are to be used. Otherwise, default values are used with z/OS console services.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a `4nn` code or a `5nn` code. The HTTP status codes are described in the topics for the individual services.

In addition, a JSON object describes the error.

Error logging

Errors from the z/OS console services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Issue a command

Use this operation to issue a command by using a system console.

HTTP method and URI path

```
PUT /zosmf/restconsoles/consoles/consolename
PUT /zosmf/restconsoles/consoles/defcn
```

where:

consolename
is the name of the EMCS console that is used to issue the command. The name must be 2 - 8 characters long, and cannot be defcn, which is reserved.

defcn
indicates that the name of the console that is used to issue the command is generated by the REST Console API, by adding CN to the logon user ID. For example, if the logon user ID is CJOEY, the console name is CJOEYCN. If the user ID is longer than 6 characters, the user ID is truncated. For example, if the user ID is ZOSMFAD, then the console name is ZOSMFACN.

Query parameters

None.

Description

This operation issues a command, based on the properties that are specified in the request body.

On successful completion, HTTP status code `200` is returned. A JSON object typically contains the command response.

When a command is issued synchronously, the console API attempts to get the solicited messages immediately after the command is issued. If there are no messages available within a certain time interval, approximately 3 seconds when your system workload is not high, the API returns "cmd-response": "" in the response body.

A value for cmd-response of the empty string, "", usually means that there is no command response. However, it is also possible that the command response arrived after 3 seconds. If that is the case, you can use the cmd-response-url field in the response body to retrieve the command response. You might do this several times to ensure that all messages related to the command are retrieved.

Alternatively, you might examine unsolicited messages, that is, additional messages that are not part of the command response. To do this, you issue the command with option unsol-key to detect a keyword in the unsolicited messages.

For the properties that you can specify on the request body, see "Request content."

For a description of the response content, see "Response content" on page 368.

Request content

The request content is expected to contain a JSON object. See Table 206 on page 365 for a description of the fields.

Table 206. Request content for the issue command request

Field name	Required or Optional	Description
cmd	Required	Specifies the command to issue.
sol-key	Optional	Specifies a keyword that you want to detect in solicited messages, that is, the command response. Case is not significant.
unsol-key	Optional	Specifies a keyword that you want to detect in unsolicited messages. Case is not significant.
detect-time	Optional	Indicates how long the console attempts to detect the value of unsol-key in the unsolicited messages. The unit is seconds. For example, if the value of detect-time is 10, the console checks the unsolicited messages for 10 seconds. The default is 30 seconds.
async	Optional	Indicates the method of issuing the command: **Y** Asynchronously **N** Synchronously. This is the default.
system	Optional	Name of the system in the same sysplex that the command is routed to. The default is the local system.
unsol-detect-sync	Optional	Indicates how to detect the keyword that is specified with the unsol-key field from unsolicited messages: **Y** Synchronously detect the keyword from unsolicited messages. The request is not returned until the unsol-detect-timeout value has elapsed or the detection result is complete. **N** Asynchronously detect the keyword from unsolicited messages. The request is returned immediately with the detection-url. The client application must invoke the value of detection-url to poll the result of the detection asynchronously. This is the default is the field is not specified.
unsol-detect-timeout	Optional	Indicates how long, in seconds, the request is blocked when the value for unsol-detect-sync is Y and the detection result has not been completed. The default value, 20 seconds, is used when this field is not specified and the value for unsol-detect-sync is Y.

Authorization requirements

See “Required authorizations” on page 362.

HTTP status codes

For a successful request, HTTP status code `200` is returned and the response body is provided, as described in “Request content” on page 364.

For unsuccessful requests, the service returns the status codes that are described in Table 207 on page 366.

Table 207. HTTP error response codes for an issue command request

HTTP Status	Return Code	Reason Code	Reason	Description
400	1	3	No match for method PUT and pathInfo=' %s '.	The path information, %s, in the original request contains a URL that is not acceptable for the z/OS Console API. Ensure that the request contains the correct URL. A console name must be 2 - 8 alphanumeric characters, the first of which must be alphabetic or one of the special characters #, $ or @.
400	1	6	The Content-Type ' %s ' cannot be handled, 'application/json' is expected.	The Content-Type, %s, in the original request contains an invalid value for the HTTP Content-Type header. The z/OS Console API accepts only application/json for the Content-Type. Update the value of the HTTP Content-Type header and make sure that the request body is in JSON format.
400	1	11	Format of parameter 'rsize' is wrong, it cannot be changed to a number.	The rsize parameter requires a numeric value, but the supplied value is not a number. Change the value to a number.
400	1	12	The body of the request is not in JSON format.	The request body must be in JSON format, but the supplied request body is not in JSON format. Correct the request body to be in JSON format.
400	1	13	Cannot find 'cmd' in request body, or value of 'cmd' is empty. No command will be issued.	No cmd field was found in the request body, or the cmd field is empty. The cmd field specifies the command to be issued. No command is issued. Ensure the request body includes a cmd field with a value.
400	1	14	Invalid console name. The length of console name must be greater than 1 and less than 9.	The console name that is specified in the URL is not valid. Supply a valid console name.
400	1	17	Command length must be less than 127.	The value of the cmd field exceeds the maximum length of a command, which is 126 characters. Please provide an valid command.
400	1	18	Command is invalid.	The command that was issued is invalid. Please provide an valid command.
500	2	7	Internal https connection timeout.	The internal connection to the zOSMF REST TSO service timed out. Retry the request. If the problem persists, contact your zOSMF administrator.
500	2	8	I/O error when connecting TSO service	An error occurred in the internal connection to the zOSMF REST TSO service. Retry the command. If the problem persists, contact your zOSMF administrator.
500	3	1	REST TSO service returned non-200 status code when creating TSO address space.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when creating a TSO address space. Contact your zOSMF administrator.

Table 207. HTTP error response codes for an issue command request (continued)

HTTP Status	Return Code	Reason Code	Reason	Description
500	3	2	REST TSO service returned an error message when creating a TSO address space.	The internal connection to the zOSMF REST TSO service returned a success (200) HTTP response with an unexpected message. Contact your zOSMF administrator.
500	3	3	REST TSO service returned non-200 status code when setting up solicited and unsolicited message display.	The attempt to prepare a TSO address space failed. Retry the request. If the problem persists, contact your zOSMF administrator.
500	3	4	Cannot retrieve TSO AS key from data returned by REST TSO service.	The attempt to prepare a TSO address space failed. Retry the request. If the problem persists, contact your zOSMF administrator
500	3	6	Unknown error occurred when creating or getting the TSO AS.	An unknown error occurred during an attempt to create a TSO address space. Retry the request. If the problem persists, contact your zOSMF administrator.
500	3	7	REST TSO service returned a non-200 status code.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when issuing a command. Contact the zOSMF administrator.
500	3	8	Server end program cannot be found.	The server end program of the REST Console API cannot be found. Contact the zOSMF administrator.
500	3	9	JSON serialization failed when calling a REXX program.	An internal error occurred during the process of translating the response from a TSO service. Contact the zOSMF administrator.
500	3	10	Unexpected messages were found when calling a REST TSO service.	TSO error messages were found when calling the REST TSO service to issue a command. Contact the zOSMF administrator.
500	5	1	REST TSO service returned a non-200 status code when creating a console.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when creating a console. Contact the zOSMF administrator.
500	5	2	Invalid parameters were passed in when creating a console object.	An internal error occurred during an attempt to create a console. Contact the zOSMF administrator.
500	5	3	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	4	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	5	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	6	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.

Table 207. HTTP error response codes for an issue command request (continued)

HTTP Status	Return Code	Reason Code	Reason	Description
500	5	7	Create console failed due to a TSO console command error.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	8	The number of consoles has reached the limit.	The maximum number of consoles supported by the z/OS Console API was reached. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	8	13	Recovery of persistence data is not complete, try later.	The z/OS Console API recovery process was not complete when you issued the request. Wait a few seconds, then retry the request.
500	8	14	Cannot get the command response.	The z/OS Console API failed to get the command response. Retry the request. If the problem persists, contact the z/OSMF administrator.

Response content

On successful completion, the service returns a response body, which contains a JSON object. The JSON object varies depending on whether the request was synchronous or asynchronous. For a description of fields in the JSON object, see Table 208 and Table 209.

Table 208. Response content for a successful synchronous issue command request

Field name	Description
cmd-response	Command response.
cmd-response-url	URL that can be used to retrieve the command response later when the value for cmd-response is empty.
cmd-response-uri	URI that can be used to retrieve the command response later when the value for cmd-response is empty. The URI starts with /zosmf.
cmd-response-key	Key that can be used to retrieve the command response later when the value for cmd-response is empty.

Table 209. Response content for a successful asynchronous issue command request

Field name	Description
cmd-response-url	URL that can be used to retrieve the command response.
cmd-response-uri	URI that can be used to retrieve the command response. The URI starts with /zosmf.
cmd-response-key	Key that can be used to retrieve the command response.
sol-key-detected	Returned when sol-key was specified, and unsol-detect-sync was specified as N or not specified. If the keyword was detected in the command response, the value is true. Otherwise, the value is false.
detection-url	The URL that can be used later to retrieve the detection result for detecting a keyword from unsolicited messages. Returned when unsol-key was specified to detect a keyword in unsolicited messages, and unsol-detect-sync was specified as N or not specified.
detection-uri	The URI that can be used later to retrieve the detection result for detecting a keyword from unsolicited messages. Returned when unsol-key was specified to detect a keyword in unsolicited messages, and unsol-detect-sync was specified as N or not specified.

Table 209. Response content for a successful asynchronous issue command request (continued)

Field name	Description
detection-key	Returned when unsol-key was specified to detect a keyword in unsolicited messages. You can use this to retrieve the result.
status	Status of the unsolicited detection request. Returned when sol-key was specified, and unsol-detect-sync is specified as Y. The values are: **expired** The detection request has expired. No matching record in the unsolicited messages was found in the time specified by detect-time. **detected** Matching records in the unsolicited messages were found in the time specified by detect-time. msg contains the message that contains the keyword. **timeout** The unsol-detect-timeout elapsed before the detection result completed. **detection-url** The URL that could be used to retrieve the detection result for detecting a keyword from unsolicited messages. **detection-uri** The URI that could be used to retrieve the detection result for detecting a keyword from unsolicited messages. **detection-key** The key that could be used to retrieve the unsolicited keyword detection result.

The client application can use any one of detection-url, detection-uri, ordetection-key to retrieve the detection result.

If a failure occurs, the response body contains a JSON object that describes the error.

Table 210. Response content for an unsuccessful issue command request

Field name	Description
return-code	Category of the error.
reason-code	Specific error.
reason	Text that describes the cause of the error.

Example HTTP interactions

1. The example in Figure 208 shows a request to issue the system command d a,pegasus synchronously.

```
PUT https://pev061.pok.ibm.com/zosmf/restconsoles/consoles/ibmusecn

{"cmd":"d a,PEGASUS"}
```

Figure 208. Sample request to issue a command synchronously

The following is the response body for the request. In the response, \r is the return character.

```
{"cmd-response": "IEE115I 07.30.59 2016.011 ACTIVITY 070\r JOBS  MVS   TS USERS   SYSAS   INITS  ACTIVEVMAX
VTAM   OAS\r 00003  00013   00002   00032  00011   00001\V00020   00015\r PEGASUS NOT FOUND",
"cmd-response-uri":"\/zosmf\/restconsoles\/consoles\/ibmusecn\/solmsgs\/C005291",
"cmd-response-url":"https:\/\/pev061.pok.ibm.com:443\/zosmf\/restconsoles\/consoles\/ibmusecn\/solmsgs\/C005291",
"cmd-response-key":"C005291"}
```

Figure 209. Sample response body

2. The example in Figure 210 shows a request to issue the system command d a,PEGASUS synchronously, and attempt to detect PEGASUS in the command response.

```
PUT https://pev076.pok.ibm.com/zosmf/restconsoles/consoles/ibmusecn

{"cmd":"d a,PEGASUS","sol-key":"PEGASUS"}
```

Figure 210. Sample request to issue a command and detect a keyword

The following is the response body for the request.

```
{"cmd-response": "IEE115I 07.30.59 2016.011 ACTIVITY 070\r JOBS  MVS   TS USERS   SYSAS   INITS  ACTIVEVMAX
VTAM   OAS\r 00003  00013   00002   00032  00011   00001\V00020   00015\r PEGASUS NOT FOUND",
"sol-key-detected":true, "cmd-response-uri":"\/zosmf\/restconsoles\/consoles\/ibmusecn\/solmsgs\/C005291",
"cmd-response-url":"https:\/\/pev076.pok.ibm.com:443\/zosmf\/restconsoles\/consoles\/ibmusecn\/solmsgs\/C005291",
"cmd-response-key":"C005291"}
```

Figure 211. Sample response body

3. The example in Figure 212 shows a request to issue the system command s PEGASUS asynchronously and attempt to detect PEGASUS in the unsolicited messages.

```
PUT https://pev076.pok.ibm.com/zosmf/restconsoles/consoles/defcn

{"cmd":"s PEGASUS","unsol-key":"PEGASUS","async":"Y"}
```

Figure 212. Sample request to issue a system command asynchronously

The following is the response body for the request.

```
{"cmd-response-uri":"\/zosmf\/restconsoles\/consoles\/ibmusecn\/solmsgs\/C005291",
"detection-url":"https:\/\/pev076.pok.ibm.com:443\/zosmf\/restconsoles\/consoles\/ibmusecn\/detections\/dec6800",
"detection-uri":"\/zosmf\/restconsoles\/consoles\/ibmusecn\/detections\/dec6800",
"detection-key":"dec6800",
"cmd-response-url":"https:\/\/pev076.pok.ibm.com:443\/zosmf\/restconsoles\/consoles\/ibmusecn\/solmsgs\/C005291",
"cmd-response-key":"C005291"}
```

Figure 213. Sample response body

4. The example in Figure 214 shows a request to issue an s PEGASUS command synchronously, using the default console, and detect keyword PEGASUS in the unsolicited messages synchronously. The keyword is found in unsolicited messages before the timeout is reached.

```
PUT https://pev061.pok.ibm.com/zosmf/restconsoles/consoles/defcn

{"cmd":"s PEGASUS","unsol-key":"PEGASUS","unsol-detect-sync":"Y"}
```

Figure 214. Sample request to issue an s PEGASUS command synchronously and detect the keyword PEGASUS

The following is the response body for the request.

```
{"status":"detected","cmd-response":"BPXM023I (ZOSMFAD) CFZ02300I: Configuration property
httpAuthType is not supported. Setting ignored.","msg":"$HASP100 PEGASUS ON STCINRDR"}
```

Figure 215. Sample response body

5. The example in Figure 214 on page 370 shows a request to issue an s PEGASUS command asynchronously, using the default console, and detect keyword XIAOX in the unsolicited messages synchronously. The detection result is not complete before the timeout (the default of 20 seconds) was reached.

```
PUT https://pev061.pok.ibm.com/zosmf/restconsoles/consoles/defcn

{"cmd":"s PEGASUS","async":"Y","unsol-key":"XIAOX","unsol-detect-sync":"Y"}
```

Figure 216. Sample request to issue an s PEGASUS command asynchronously and detect the keyword XIAOX

The following is the response body for the request.

```
{"cmd-response-uri":"\/zosmf\/restconsoles\/consoles\/defcn\/solmsgs\/C2790426","detection
-url":"https:\/\/pev061.pok.ibm.com:443
\/zosmf\/restconsoles\/consoles\/defcn\/detections\/D5303564","detection-uri":"
\/zosmf\/restconsoles\/consoles\/defcn\/detections
\/D5303564","detection-key":"D5303564","status":"timeout","cmd-response-url":"https:
\/\/pev061.pok.ibm.com:443\/zosmf\/restconsoles\/consoles\/defcn\/solmsgs\/C2790426",
"cmd-response-key":"C2790426"}
```

Figure 217. Sample response body

Get a command response

Use this operation to get the response to a command that was issued asynchronously with the Issue Command service.

HTTP method and URI path

```
GET /zosmf/restconsoles/consoles/console-name/solmsgs/Ckey-number
GET /zosmf/restconsoles/consoles/defcn/solmsgs/Ckey-number
```

where:

consolename
is the name of the EMCS console that was used in the Issue Command request.

defcn
indicates that name of the console that was used to issue the command was generated by the REST Console API.

Ckey-number
is the command response key from the Issue Command request.

The URL is returned by the Issue Command request in the cmd-response-url field.

Query parameters

None.

Description

This operation gets the messages that were issued in response to a command that was issued asynchronously with the Issue Command service. For the properties that you can specify, see "Request content."

On successful completion, HTTP status code `200` is returned. The response content is described in "Response content" on page 374.

The Issue Command service returns the URL of the command response in the cmd-response-url field. For more information about the response content of the Issue Command service, see "Response content" on page 368.

Request content

None.

Authorization requirements

See "Required authorizations" on page 362.

HTTP status codes

On successful completion, HTTP status code `200` is returned and the response body is provided, as described in "Response content" on page 374.

Otherwise, the HTTP status codes in Table 211 on page 373 are returned for the indicated errors.

Table 211. HTTP error response codes for a get command response request

HTTP Status	Return Code	Reason Code	Reason	Description
400	1	3	No match for method GET and pathInfo=' %s '.	The path information, *%s*, in the original request contains a URL that is not acceptable for the z/OS Console API. Ensure that the request contains the correct URL. A console name must be 2 - 8 alphanumeric characters, the first of which must be alphabetic or one of the special characters #, $ or @.
400	1	14	Invalid console name. The length of console name must be greater than 1 and less than 9.	The console name that is specified in the URL is not valid. Supply a valid console name.
400	1	18	Command is invalid.	The command that was issued is invalid. Please provide a valid command.
500	3	1	REST TSO service returned a non-200 status code when creating a TSO address space.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when creating a TSO address space. Contact your zOSMF administrator.
500	3	2	REST TSO service returned an error message when creating a TSO address space.	The internal connection to the zOSMF REST TSO service returned a success (200) HTTP response with an unexpected message. Contact your zOSMF administrator.
500	3	3	REST TSO service returned non-200 status code when setting up solicited and unsolicited message display.	The attempt to prepare a TSO address space failed. Retry the request. If the problem persists, contact your zOSMF administrator.
500	3	4	Cannot retrieve TSO AS key from data returned by REST TSO service.	The attempt to prepare a TSO address space failed. Retry the request. If the problem persists, contact your zOSMF administrator
500	3	7	REST TSO service returned a non-200 status code.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when issuing a command. Contact the zOSMF administrator.
500	3	8	Server end program cannot be found.	The server end program of the REST Console API cannot be found. Contact the zOSMF administrator.
500	3	9	JSON serialization failed when calling a REXX program.	An internal error occurred during the process of translating the response from a TSO service. Contact the zOSMF administrator.
500	3	10	Unexpected messages were found when calling a REST TSO service.	TSO error messages were found when calling the REST TSO service to issue a command. Contact the zOSMF administrator.
500	5	1	REST TSO service returned a non-200 status code when creating a console.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when creating a console. Contact the zOSMF administrator.
500	5	2	Invalid parameters were passed in when creating a console object.	An internal error occurred during an attempt to create a console. Contact the zOSMF administrator.

Table 211. HTTP error response codes for a get command response request (continued)

HTTP Status	Return Code	Reason Code	Reason	Description
500	5	3	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	4	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	5	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	6	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	7	Create a console failed due to a TSO console command error.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	8	The numbers of consoles has reached the limit.	The maximum number of consoles supported by the z/OS Console API was reached. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	8	13	Recovery of persistence data is not complete, try later.	The z/OS Console API recovery process was not complete when you issued the request. Wait a few seconds, then retry the request.
500	10	1	The message you requested cannot be retrieved due to earlier shutdown of zOSMF server.	The z/OS Console API failed to get the command response. Retry the request. If the problem persists, contact the z/OSMF administrator.

Response content

On successful completion, the service returns a response body, which contains a JSON object. Table 212 lists the fields in the JSON object.

Table 212. Response content for a successful get command response request

Field name	Description
cmd-response	Command response
sol-key-detected	Returned when sol-key was specified on the Issue Command service. If the keyword specified with sol-key was found in the command response, the value is true. Otherwise, the value is false.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 213. Response content for an unsuccessful get command response request

Field name	Description
return-code	Category of the error.

Table 213. Response content for an unsuccessful get command response request (continued)

Field name	Description
reason-code	Specific error.
reason	Text that describes the cause of the error.

Example HTTP interaction

The example in Figure 218 shows a request to get the response to a system command that was issued asynchronously. The command was issued with a generated console name. The command response key returned by the issue command request is C003715.

```
GET https://pev061.pok.ibm.com:443/zosmf/restconsoles/consoles/ibmusecn/solmsgs/C508135
```

Figure 218. Sample request to get the response for a system command that was issued asynchronously

The following is the response body for the request.

```
{"cmd-response":"IEE215I 07.36.34 2016.011 PARMLIB DISPLAY 513\R PARMLIB DATA SETS SPECIFIED\R AT IPL\R ENTRY FLAGS
VOLUME DATA SET\R 1  S  PEVTS3 CIMSSRE.R22ONLY.PARMLIB\r  2   S  PEVTS3
CIMSSRE.R14ONLY.PARMLIB\r  3  S  PEVTS3 CIMSSRE.R13ONLY.PARMLIB\r  4  S PEVTS3
CIMSSRE.R12ONLY.PARMLIB\r  5  S  PEVTS3 CIMSSRE.PARMLIB\r  6  S  PEVTST HDENNIS..ZOS17.PARMLIB\r
7  S  CTTPAK XESCT.PARMLIB\r  8  S  CTTPAK SYS1.PARMLIB\r  9  S  SDR22 SYS1.PARMLIB.POK\r
S  SDR22 SYS1.PARMLIB.INSTALL"}
```

Figure 219. Sample response body for a get command response request

Get the detect result for unsolicited messages

Use this operation to get the result for detecting a keyword in unsolicited messages after an Issue Command request. The command must have been issued with the unsol-key field.

HTTP method and URI path

```
GET /zosmf/restconsoles/consoles/consolename/detections/Dkey-number
GET /zosmf/restconsoles/consoles/defcn/detections/Dkey-number
```

where:

consolename
is the name of the EMCS console that was used in the Issue Command request.

defcn
indicates that name of the console that was used to issue the command was generated by the REST Console API.

Dkey-number
is the detection key from the Issue Command request.

The URL is be returned by the Issue Command request in the detection-url field.

Query parameters

None.

Description

This operation gets the results of attempting to detect a keyword in the unsolicited messages that were issued following an Issue Command request. The keyword being detected was specified with the unsol-key field on the Issue Command service.

On successful completion, HTTP status code `200` is returned. The response content is described in "Response content" on page 378.

The Issue Command service returns the URL in the detection-url field. For more information about the response content of the Issue Command service, see "Response content" on page 368.

Request content

None.

Authorization requirements

See "Required authorizations" on page 362.

HTTP status codes

On successful completion, HTTP status code `200` is returned and the response body is provided, as described in "Response content" on page 378.

Otherwise, the HTTP status codes in Table 214 on page 377 are returned for the indicated errors.

Table 214. HTTP error response codes for a detect result for unsolicited messages request

HTTP Status	Return Code	Reason Code	Reason	Description
400	1	3	No match for method GET and pathInfo=' %s '.	The path information, *%s*, in the original request contains a URL that is not acceptable for the z/OS Console API. Ensure that the request contains the correct URL. A console name must be 2 - 8 alphanumeric characters, the first of which must be alphabetic or one of the special characters #, $ or @.
400	1	14	Invalid console name. The length of console name must be greater than 1 and less than 9.	The console name that is specified in the URL is not valid. Supply a valid console name.
500	3	1	REST TSO service returned a non-200 status code when creating a TSO address space.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when creating a TSO address space. Contact your zOSMF administrator.
500	3	2	REST TSO service returned an error message when creating a TSO address space.	The internal connection to the zOSMF REST TSO service returned a success (200) HTTP response with an unexpected message. Contact your zOSMF administrator.
500	3	3	REST TSO service returned a non-200 status code when setting up solicited and unsolicited message display.	The attempt to prepare a TSO address space failed. Retry the request. If the problem persists, contact your zOSMF administrator.
500	3	4	Cannot retrieve TSO AS key from data returned by REST TSO service.	The attempt to prepare a TSO address space failed. Retry the request. If the problem persists, contact your zOSMF administrator
500	3	7	REST TSO service returned a non-200 status code.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when issuing a command. Contact the zOSMF administrator.
500	3	8	Server end program cannot be found.	The server end program of the REST Console API cannot be found. Contact the zOSMF administrator.
500	3	9	JSON serialization failed when calling a REXX program.	An internal error occurred during the process of translating the response from a TSO service. Contact the zOSMF administrator.
500	3	10	Unexpected messages were found when calling a REST TSO service.	TSO error messages were found when calling the REST TSO service to issue a command. Contact the zOSMF administrator.
500	5	1	REST TSO service returned a non-200 status code when creating a console.	The internal connection to the zOSMF REST TSO service returned an error HTTP response when creating a console. Contact the zOSMF administrator.
500	5	2	Invalid parameters were passed in when creating a console object.	An internal error occurred during an attempt to create a console. Contact the zOSMF administrator.

Table 214. HTTP error response codes for a detect result for unsolicited messages request (continued)

HTTP Status	Return Code	Reason Code	Reason	Description
500	5	3	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	4	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	5	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	6	Cannot retrieve local time zone.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	7	Create console failed due to TSO console command error.	An internal error occurred during an attempt to prepare a console. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	8	The number of consoles has reached the limit.	The maximum number of consoles supported by the z/OS Console API was reached. Retry the request. If the problem persists, contact the z/OSMF administrator.
500	5	9	Cannot find the result for specified detection ID.	Cannot find the result for the specified detection ID. Ensure that the detection ID is correct.
500	8	13	Recovery of persistence data is not complete, try later.	The z/OS Console API recovery process was not complete when you issued the request. Wait a few seconds, then retry the request.
500	10	2	The detection result you requested cannot be retrieved due to earlier shutdown of the z/OSMF server.	The detection result cannot be retrieved because of an earlier shutdown of zOSMF server.

Response content

On successful completion, the service returns a response body, which contains a JSON object. Table 215 on page 379 lists the fields in the JSON object.

Table 215. Response content for a successful get detect result request

Field name	Description
status	Status of the detection request: **waiting** The detection request is still valid, the keyword has not yet been detected in the unsolicited messages. **expired** The detection request expired, and the keyword was not found in the unsolicited messages. The detection request expires when the value for detect-time on the issue command request is exceeded. **detected** The keyword was found in the unsolicited messages.
msg	Returned when the value of status is detected. This is the message that contains the keyword that was detected.

If a failure occurs, the response body contains a JSON object with a description of the error.

Table 216. Response content for an unsuccessful get detect result request

Field name	Description
return-code	Category of the error.
reason-code	Specific error.
reason	Text that describes the cause of the error.

Example HTTP interaction

1. The example in Figure 220 shows a request to get the results for a detect request. The command was issued with a generated console name. The detection key returned by the issue command request is D002185.

```
GET https://pev076.pok.ibm.com/zosmf/restconsoles/consoles/defcn/detections/D002185
```

Figure 220. Sample request to get the detect result

The following is the response body for the request. The request is still valid, but the keyword has not been found.

```
{"status":"waiting","msg":""}
```

Figure 221. Sample response body for a get detect result request

2. The example in Figure 220 shows a request to get the results for a detect request. The command was issued with a generated console name. The detection key that was returned by the issue command request is D122033.

```
GET https://pev076.pok.ibm.com/zosmf/restconsole/consoles/defcn/detections/D122033
```

Figure 222. Sample request to get the detect result

The following is the response body for the request. The keyword was found. In the response, \r is the return character.

```
{"status":"detected","msg":"BPXM023I (ZOSMFAD)\r CFZ12584W: CIM Runtime Environment Userid currently only has READ\r access to
BPX.SERVER. It is recommended to have either UPDATE access\r to BPX.SERVER or has to be UID 0."}
```

Figure 223. Sample response body for a successful get detect result request

z/OS data set and file REST interface

The z/OS data set and file REST interface is an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. A set of REST services is provided for working with data sets and UNIX files on a z/OS system.

The z/OS data set and file REST interface services provide a programming interface for working with z/OS data sets and UNIX files. This function is similar to using GET and PUT requests through file transfer protocol (FTP), but secured through traditional z/OS security controls for user authentication and resource authorizations. For setup details, see "Required authorizations" on page 384.

Table 217 lists the operations that the z/OS data set and file REST interface services provide.

Table 217. Operations provided through the z/OS data set and file REST interface services.

Operation	HTTP method and URI path
"List the z/OS data sets on a system" on page 386	GET /zosmf/restfiles/ds?dslevel=<*dataset_name_pattern*>[&volser=<*volser*>*&start=*<*dsname*>]
"List the members of a z/OS data set" on page 389	GET /zosmf/restfiles/ds/<*dataset_name*>/member?start=<member>&pattern=<mem-pat>
"Retrieve the contents of a z/OS data set or member" on page 392	GET /zosmf/restfiles/ds/[-(<*volser*>)/]<*data-set-name*>[(<*member-name*>)]
"Write data to a z/OS data set or member" on page 398	PUT /zosmf/restfiles/ds/[-(<*volser*>)/]<*data-set-name*>[(<*member-name*>)]
"Create a sequential and partitioned data set" on page 403	POST /zosmf/restfiles/ds/<*data-set-name*>
"Delete a sequential and partitioned data set" on page 406	DELETE /zosmf/restfiles/ds/<*data-set-name*> DELETE /zosmf/restfiles/ds/*-(*<*volume*>*)/*<*data-set-name*>
"Delete a partitioned data set member" on page 408	DELETE /zosmf/restfiles/ds/<*dataset-name*>*(*<*member-name*>*)*
"z/OS Data set and member utilities" on page 410	PUT /zosmf/restfiles/ds/<*to-data-set-name*>
"Access Method Services Interface" on page 414	PUT /zosmf/restfiles/ams
"List the files and directories of a UNIX file path" on page 416	GET /zosmf/restfiles/fs?path=<*file-path-name*>
"Retrieve the contents of a z/OS UNIX file" on page 419	GET /zosmf/restfiles/fs/<*file-path-name*>

Table 217. Operations provided through the z/OS data set and file REST interface services. (continued)

Operation	HTTP method and URI path
"Write data to a z/OS UNIX file" on page 423	PUT /zosmf/restfiles/fs/<*filepath-name*>
"Create a UNIX file or directory" on page 426	POST /zosmf/restfiles/fs/<*file-path-name*>
"Delete a UNIX file or directory" on page 429	DELETE /zosmf/restfiles/fs/<*file-pathname*>
"z/OS UNIX file utilities" on page 431	PUT /zosmf/restfiles/fs/<*file-path-name*>
"List z/OS UNIX Filesystems" on page 437	GET /zosmf/restfiles/mfs/
"Create z/OS UNIX zFS Filesystem" on page 439	POST /zosmf/restfiles/mfs/zfs/<*file-system-name*>
"Delete z/OS UNIX zFS Filesystem" on page 441	DELETE /zosmf/restfiles/mfs/zfs/<*file-system-name*>
"Mount a UNIX file system" on page 442	PUT /zosmf/restfiles/mfs/<*file-system-name*>
"Unmount a UNIX file system" on page 444	PUT /zosmf/restfiles/mfs/<*file-system-name*>

Processing overview

The z/OS data set and file REST interface services can be invoked by any client application, running on the local z/OS system or a remote system. Your program (the client) initiates a request to the server through a standard HTTP request method, such as GET or PUT. If the server determines that the request is valid, it performs the requested service and returns an HTTP response to your program.

For a successful request, this response takes the form of an HTTP 2*nn* status code and, if applicable, a result set that is passed back to your program. Depending on which service is requested, the result set might be returned in a format that requires parsing by your program, such as a JSON object. In other cases, the results might be returned in another format, such as plain text or binary data.

For an unsuccessful request, the server response consists of a non-OK HTTP response code and details of the error, which are provided in the form of a JSON object.

The contents of the JSON objects are described in "JSON document specifications for z/OS data set and file REST interface requests" on page 446.

Note:

If the URL contains a resolved character such as # $ @, it must be URL-encoded, so that it can be escaped.
For example:
GET /zosmf/restfiles/ds/SYS1.PROCLIB(#ABC) HTTP/1.1
Should be changed to:
GET /zosmf/restfiles/ds/SYS1.PROCLIB(%23ABC) HTTP/1.1

Common HTTP Request Headers

X-IBM-Async-Threshold = <nnn>

This header can be added to a request to enable support for asynchronous responses with the HTTP status code `202 Accepted`. This specifies the number of seconds that the client wishes to wait for a response before receiving a `202 Accepted` response. This response includes a Location response header with the URL (excluding protocol, host, and port) that can be used with a subsequent GET method request to obtain the results from the original request. Each subsequent request includes its own X-IBM-Async-Threshold header if additional async responses are required. The value of this header must be an integer between 0 and 60 seconds. A value of 0 indicates that an async response is returned if the actual response is not immediately available. If X-IBM-Async-Threshold is specified, then X-IBM-Response-Timeout does not apply and is ignored. A DELETE method request can also be sent to the URL returned in an asynchronous response to abandon the original request and terminate the associated CEA TSO address space.

Example of an Asynchronous request

Request:

```
GET  /zosmf/restfiles/ds?dslevel=D10 HTTP/1.1
X-IBM-Async-Threshold: 3
```

Response:

```
202 Accepted
X-Powered-By:  Servlet/3.0
Location:  /zosmf/restfiles/queue/FS11fae7
X-IBM-Txid:  tx000000000000D159
Content-Language:  en-US
Content-Length:  0
Date:  Fri, 18 Nov 2016 07:17:21 GMT
```

Request:

```
GET  GET /zosmf/restfiles/queue/FS11fae7 HTTP/1.1 HTTP/1.1
X-IBM-Async-Threshold: 3
```

Response:

```
200 OK
Content-Type:  application/json; charset=UTF-8
Content-Length:  39199
X-IBM-Txid:  tx000000000000D160
Date:  Fri, 18 Nov 2016 07:18:32 GMT
{"items":[
  {"dsname":"D10"},
  {"dsname":"D10.$DATA.SETS"},
  {"dsname":"D10.AAAAA"},
  {"dsname":"D10.AACTIVE.JCL"},
  {"dsname":"D10.AA11797.R1K.D050701A"},
  {"dsname":"D10.AA12484.HDZ11K0"},
  {"dsname":"D10.AA12484.HDZ11K0.TRSD"},
  {"dsname":"D10.ABACKUP.SM02631.FPGA.D14163.T131433"},
  {"dsname":"D10.ABARS1.C.C01V0001"},
```

X-IBM-Response-Timeout = <nnn>

This specifies the number of seconds that a TsoServlet request runs before a timeout occurs and an exception is returned to the client. This time does not include the time that can be required to start a new CEATSO address space. The default is 30 seconds, and the allowed range for this value is between 5 and 600 seconds. An invalid value supplied for this header is converted into the closest valid value and the request proceeds.

Note: This timeout does not affect any timeouts that may occur in the z/OSMF Websphere container or the REST service client.

Common HTTP Response Headers

X-IBM-Txid = <string>

This header returns the transaction id that was assigned by z/OSMF to the request. It can be useful for diagnostic purposes to identify z/OSMF log records relating to a failed transaction. This txid will also be logged in the TSO address space. It should not be used for other purposes; its format can change; and it may or may not be present in future releases.

Specifying an entity tag with your read and write requests

During request processing, your program's access to the resource (a data set or file) is serialized by z/OS. No other users can read the resource or write to it, thus preventing concurrent updates of the resource from overwriting each other. Serialization, however, ends with the completion of the request. If your program must perform multiple read and write requests on a resource, you require a method of ensuring that the resource is not modified between your program's requests. Otherwise, you might overlay another user's changes.

To help you to ensure the integrity of a resource between requests, the z/OS data set and file REST interface supports the use of an *entity tag* (or *ETag*) on your requests. Obtained on the initial read (GET) request, the ETag is an identifier that is assigned by the web server to a specific version of the resource. If the resource content changes, a new ETag is associated with the resource.

To determine whether a resource was changed between requests, your program supplies its ETag as a header value on each request. If the ETag matches the current ETag for the resource, the system considers the resource to be unchanged and performs the request. Otherwise, the system fails the request; your program must obtain the ETag again before it can perform the request.

Generally, the process of updating the contents of a z/OS data set or UNIX file is as follows:

1. Retrieve the current content of the resource by using a GET request.
2. The server returns the contents of the resource in the response body as plain text, along with information about the resource, in the response headers:
   - Content-Length header specifies the length of the data that was returned
   - ETag header specifies the ETag that identifies the current version of the resource.
3. Replace the contents of the resource by using a PUT request. The request includes the following headers:
   - A request header to supply the ETag that was returned from the previous GET request on that resource. If the token still matches, the resource was not changed since the previous GET request. If the supplied token does not match a currently valid token, the PUT request fails with an `HTTP 412`. This response indicates that the host system file was modified in the time since the read operation was performed.
   - Optionally, a request header to specify whether data conversion is required.

   For a PUT request, the request body contains the new contents of the file.

   After the data is written, the `204 No Content` response is returned with a new ETag, for use with any subsequent read or write requests.

Suppose you only want to replace a resource with a new copy, without first reading the contents of the current resource. To overlay the resource, have your program issue the initial GET request to obtain the ETag. Here, you would specify a maximum read amount of 0. Then, have your program issue a PUT request with the ETag and the new data to be written to the resource.

Content type used for HTTP request and response data

The JSON content type (`"Content-Type: application/json"`) is used for request and response data. For the detailed format of each JSON object, see "JSON document specifications for z/OS data set and file REST interface requests" on page 446.

Required installation

To enable the z/OS data set and file REST interface services, IBM supplies a default procedure in your z/OSMF order, which you must install before you configure z/OSMF. For information, see *IBM z/OS Management Facility Configuration Guide*.

Required authorizations

Generally, your user ID requires the same authorizations for using the z/OS data set and file REST interface services as when you perform these operations through a TSO/E session on the system. For example, listing the members of a z/OS data set through the z/OS data set and file REST interface requires authorization to start TSO on the system and access to the specified data set.

In addition, your user ID requires authorization to the z/OSMF SAF profile prefix on the target z/OS system, as follows:

- READ access to <*IZU_SAF_PROFILE_PREFIX*> in the APPL class.
- READ access to the <*IZU_SAF_PROFILE_PREFIX*>.izuUsers profile in the EJBROLE class.

By default, the z/OSMF SAF profile prefix is IZUDFLT.

Where applicable, further authorization requirements are noted in the descriptions of the individual z/OS data set and file REST interface services.

For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a `4nn` code or a `5nn` code. For example, `HTTP/1.1 400 Bad Request` or `HTTP/1.1 500 Internal Server Error`.

In addition, some errors might also include a returned JSON object that contains a message that describes the error. You can use this information to diagnose the error or provide it to IBM Support, if required. For the contents of the error report document, see "Error report document" on page 454.

The following HTTP status codes are valid:

HTTP 200 OK
: Request was processed successfully.

HTTP 204 No content
: Request was processed successfully, however, no content was returned. This status is normal for some types of requests, such as when no data sets or files match the filter criteria, or the specified partitioned data set has no members.

HTTP 206 Partial content
: Request was processed successfully, however, only a portion of the available content was received. The request contained the X-IBM-Max-Items header, which limited the amount of content that was returned.

HTTP 304 Not Modified
: An ETag token was included in the request. z/OSMF determined that the requested resource did not change since the ETag token was created.

HTTP 400 Bad request
: Request could not be processed because it contains a syntax error or an incorrect parameter.

HTTP 401 Unauthorized
: Request could not be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both, or the client did not authenticate to z/OSMF.

HTTP 404 Not found
Requested resource does not exist.
HTTP 405 Method not allowed
Requested resource is a valid resource, but an incorrect method was used to submit the request. For example, the request used the POST method when the GET method was expected.
HTTP 412 Precondition failed
The supplied ETag token indicated that the resource was modified since the token was created. Therefore, the request failed the If-Match precondition that was specified in the header.
HTTP 413 Request entity too large
The supplied data is too large to process. Or, the requested resource is too large to return.
HTTP 429 Too many requests
The client submitted too many unsuccessful login attempts.
HTTP 500 Internal server error
Server encountered an error. See the response body for a JSON object with information about the error.
HTTP 503 Service unavailable
Server is not available.

Error logging

Errors from the z/OS data set and file REST interface services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

List the z/OS data sets on a system

You can use this operation to list the data sets on a z/OS system. You can filter the returned list of data set names through the specification of high-level qualifiers and wildcards.

HTTP method and URI path

```
GET /zosmf/restfiles/ds/?dslevel=<filter-criteria>[&start=dsname]
```

---

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- **?dslevel=<*dataset-name-pattern*>[&volser=<*volser*>&*start*=<*dsname*>]** represents the query parameters used to qualify the request, such as a data set name and, optionally, a volume serial (VOLSER).

Standard headers

None.

Custom headers

Include the following custom HTTP header with this request:

X-IBM-Max-Items
: This header value specifies the maximum number of items to return. To request that all items be returned, set this header to 0. If you omit this header, or specify an incorrect value, up to 1000 items are returned by default.

X-IBM-Attributes
: This header specifies whether the results are to include the data set base or volume attributes.

Query parameters

You can specify the following query parameter on this request:

dslevel
: The search parameter that identifies the cataloged data sets to be listed. Either the dslevel or volser parameter must be specified and can be a fully qualified data set name or a partial data set name with a filter to display a list of matches. A partial data set name can include:
  - One or more high-level qualifiers or name segments
  - One or more wildcard symbols: percent sign (%), asterisk (*), or double asterisk (**)
  - A percent sign is a single character wildcard.
  - An asterisk is any number of characters within a qualifier.
  - A double asterisk is any number of characters within any number of lower-level qualifiers.

  The parameter values must be URL-encoded, otherwise you may receive an error message. If you use the percent sign (%) as a wildcard to filter the list of data sets returned, you must enter %25 to avoid receiving this error message: `URLDecoder: Incomplete trailing escape (%) pattern.` For example:

  ```
  GET /zosmf/restfiles/ds?dslevel=sys%25d.*lib HTTP/1.1
  ```

  Notes:

  1. The length of the data set name that you specify on the request cannot exceed 44 characters. The length limit includes wildcards, which are treated as one character each. The wildcard %25 is treated as one character.
  2. The system appends the following to any filter criteria that you specify: `.**`

3. Lowercase characters are automatically folded to uppercase.

volser
A parameter that identifies the volume serials to be searched for data sets with names that match the specified **dslevel** parameter. The volume serial is one to six characters. You cannot use wildcard characters for this parameter. If you omit this parameter, the cataloged data set name is returned by default. If this parameter is specified, the data sets on the volume that match the **dslevel** pattern are returned.

start
An optional search parameter that specifies the first data set name to return in the response document. The length of the data set name that you specify cannot exceed 44 characters, and cannot contain wildcards. If the data set name is not found for the given search, then the next data set matching the search will be returned.

X-IBM-Attributes

dsname
Requests that only data set names be returned. If you omit this header, it is set to "dsname".

base
Setting the X-IBM-Attribute to `base` returns all of the basic attributes for the data set being queried. These attributes are commonly found in the ISPF List Data set panel. The `base` key is mutually exclusive with volser, and dsname.

vol
Setting the X-IBM-Attribute to `vol` returns the volume where the data set resides.

,total
The suffix `,total`, can be added to request that the "totalRows" property is returned if more data sets than the maximum requested are available.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Example request

In the following example, the GET method is used to list all of the cataloged data sets that match the partial name IBMUSER.CONFIG.*.

```
GET /zosmf/restfiles/ds?dslevel=IBMUSER.CONFIG.* HTTP/1.1
```

Example response

A sample response is shown in Figure 224 on page 388.

```
Response:
200 OK
x-powered-by:  Servlet/3.0
Content-Type:  application/json; charset=UTF-8
Content-Length:  201
Content-Language:  en-US
Date:  Mon, 23 Nov 2015 09:10:11 GMT

Response Body:
{"items":[
{"dsname":"IBMUSER.CONFIG.FS"},
{"dsname":"IBMUSER.CONFIG.FS.DATA"},
{"dsname":"IBMUSER.CONFIG.ORIG.FS"},
{"dsname":"IBMUSER.CONFIG.ORIG.FS.DATA"}
],"returnedRows":4,"JSONversion":1}
```

Figure 224. Example: list all of the data sets.

Example request

The GET method is used to list all of the cataloged data sets with specified base attributes.

```
GET /zosmf/restfiles/ds?dslevel=**&volser=PEVTS2  HTTP/1.1

Request Headers:
X-IBM-Attributes: 'base'
```

Example response

A sample response is shown in Example: List data sets with attributes on a specified volser with X-IBM-Attributes.

```
Response:
200 OK
x-powered-by:  Servlet/3.0
Content-Type:  application/json; charset=UTF-8
Content-Length:  714
Content-Language: en-US
Date:  Mon, 23 Nov 2015 09:11:46 GMT

Response Body:
{"items":[
{"dsname":"IBMUSER.CONFIG.FS","catnm":"CATPAK.MASTER.CATALOG","dsorg":"VS",
"migr":"NO","mvol":"N","vol":"*VSAM*"},
{"dsname":"IBMUSER.CONFIG.FS.DATA","blksz":"?","catnm":"CATPAK.MASTER.CATALOG",
"cdate":"2011/08/14","dev":"3390","dsorg":"VS","edate":"***None***","extx":"1",
"lrecl":"?","migr":"NO","mvol":"N","ovf":"NO","rdate":"2015/07/28","recfm":"?",
"sizex":"14250","spacu":"CYLINDERS","used":"?","vol":"CIMSSR"},
{"dsname":"IBMUSER.CONFIG.ORIG.FS","catnm":"CATPAK.MASTER.CATALOG","dsorg":"VS",
"migr":"NO","mvol":"N","vol":"*VSAM*"},
{"dsname":"IBMUSER.CONFIG.ORIG.FS.DATA","catnm":"CATPAK.MASTER.CATALOG",
"dev":"3390","migr":"NO","mvol":"N","vol":"PEVTS2"}
],"returnedRows":4,"JSONversion":1}
```

Figure 225. Example: List all of the cataloged data sets with specified base attributes.

List the members of a z/OS data set

You can use this operation to list the members of a z/OS partitioned data set.

HTTP method and URI path

```
GET /zosmf/restfiles/ds/<dataset-name>/member?start=<member>&pattern=<mem-pat>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- **/<*dataset-name*>** identifies the data set for which members are to be listed. This parameter is required and must consist of a fully qualified data set name. The length of the data set name that you specify on the request cannot exceed 44 characters. You cannot use wildcard characters for this parameter.
- **/member** indicates that member names are to be returned.

Standard headers

None.

Custom headers

Include the following custom HTTP headers with this request:

X-IBM-Max-Items
This header value specifies the maximum number of items to return. To request that all items be returned, set this header to 0. If you omit this header, or specify an incorrect value, up to 1000 items are returned by default.

X-IBM-Attributes
This header is optional.

member
A request that only member names be returned. If you omit this header, it is set to "member".

base
Setting the X-IBM-Attribute to `base` returns all of the basic attributes for the data set member being queried. These attributes are commonly found in the ISPF List Data set panel. The `base` key is mutually exclusive with member.

,total
The suffix `,total`, can be added to request that the "totalRows" property is returned if more data set members than the maximum requested are available.

X-IBM-Migrated-Recall

This header is optional; use it to specify how a migrated data set is handled. By default, a migrated data set is recalled synchronously. The following values may be specified too:

wait
This is the default value. If the data set is migrated, wait for it to be recalled before processing the request.

nowait
If the data set is migrated, request it to be recalled, but do not wait.

error
If the data set is migrated, do not attempt to recall the data set.

Query parameters

start
An optional search parameter that specifies the first member name to return in the response document. The length of the data set name that you specify cannot exceed 8 characters, and cannot contain wildcards. If the member name is not found for the given search, then the next member matching the search is returned.

pattern
An optional search parameter restricts the returned member names to only the names that match the given pattern. The syntax of this argument is the same as "pattern" parameter of the ISPF LMMLIST service.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see “Error handling” on page 384.

For a successful request, the HTTP response includes an array of data set members, each as one of the following types of JSON list document:

For errors, the HTTP response includes error information as a JSON error report document. See “Error report document” on page 454.

Example request

In the following example, the GET method is used to list all of the members of a data set.

```
GET /zosmf/restfiles/ds/SYS1.PROCLIB/member HTTP/1.1
```

Example response

A sample response is shown in List all of the members of a data set.

```
Response

200 OK
X-Powered-By: Servlet/3.0
Content-Type: application/json; charset=UTF-8
Content-Length: 235
Content-Language: en-US
Date: Tue, 24 Nov 2015 05:31:51 GMT


Response Body

{"items":[
{"member":"CREATECD"},
{"member":"SPROCLA1"},
{"member":"TESTJCL"},
{"member":"WASACR"},
{"member":"WLMCD"},
{"member":"XRACFH"},
{"member":"XRACFHT"},
{"member":"XRACFH2"}
],"returnedRows":8,"JSONversion":1}
```

Figure 226. Example: List all of the members of a data set

Example request

In the following example, the GET method is used to list all of the members of a data set with specified base attributes.

```
GET /zosmf/restfiles/ds/SYS1.PROCLIB/member HTTP/1.1

Request Headers:
X-IBM-Attributes: 'base'
```

Example response

A sample response is shown in List all of the members of a data set with specified base attributes.

```
Response

200 OK
X-Powered-By: Servlet/3.0
Content-Type: application/json; charset=UTF-8
Content-Length: 1287
Content-Language: en-US
Date: Tue, 24 Nov 2015 05:33:57 GMT


Response Body

{"items":[
{"member":"CREATECD","vers":1,"mod":0,"c4date":"2015/08/12","m4date":"2015/08/12","cnorc":22,
"inorc":22,"mnorc":0,"mtime":"05:48","msec":"43","user":"IBMUSER","sclm":"N"},
{"member":"SPROCLA1","vers":1,"mod":12,"c4date":"2009/10/16","m4date":"2014/09/18","cnorc":132,
"inorc":122,"mnorc":0,"mtime":"07:55","msec":"23","user":"IBMUSER","sclm":"N"},
{"member":"TESTJCL","vers":1,"mod":0,"c4date":"2015/07/29","m4date":"2015/07/29","cnorc":22,
"inorc":22,"mnorc":0,"mtime":"01:49","msec":"36","user":"IBMUSER","sclm":"N"},
{"member":"WASACR"},{"member":"WLMCD","vers":1,"mod":0,"c4date":"2015/08/14","m4date":"2015/08/14",
"cnorc":22,"inorc":22,"mnorc":0,"mtime":"04:44","msec":"19","user":"IBMUSER","sclm":"N"},
{"member":"XRACFH","vers":1,"mod":1,"c4date":"2005/09/26","m4date":"2005/11/03","cnorc":514,
"inorc":506,"mnorc":8,"mtime":"11:10","msec":"45","user":"HDENNIS","sclm":"N"},
{"member":"XRACFHT","vers":1,"mod":0,"c4date":"2005/11/04","m4date":"2005/11/04","cnorc":130,
"inorc":130,"mnorc":0,"mtime":"11:28","msec":"12","user":"HDENNIS","sclm":"N"},
{"member":"XRACFH2","vers":1,"mod":0,"c4date":"2005/11/04","m4date":"2005/11/04","cnorc":130,
"inorc":130,"mnorc":0,"mtime":"11:27","msec":"43","user":"HDENNIS","sclm":"N"}
],"returnedRows":8,"JSONversion":1}
```

Figure 227. Example: List all of the members of a data set with specified base attributes.

Retrieve the contents of a z/OS data set or member

You can use this operation to retrieve the contents of a sequential data set, or a member of a partitioned data set (PDS or PDSE). To retrieve the contents of an uncataloged data set, include the volume serial on the request.

HTTP method and URI path

```
GET /zosmf/restfiles/ds/[-(<volser>)/]<dataset-name>[(<member-name>)]
```

---

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- ***-(<volser>)*** represents a volume serial. For an uncataloged data set, include this parameter to identify the volume to be searched for data sets or members that match the specified *<data-set-name>* or *<member-name>*. The length of the volume serial cannot exceed six characters. You cannot use wildcard characters for this parameter. Indirect volume serials are not supported.
- ***<datase-name>*** identifies the data set to be read. This parameter is required and must consist of a fully qualified data set name. The length of the data set name that you specify on the request cannot exceed 44 characters.
- ***<member-name>*** identifies the name of the PDS or PDSE member to be read. Include this parameter for a member read request.

Based on the object to be read, you can specify one of the following parameter combinations:

- **/*<data-set-name>***: To retrieve data from a sequential data set.
- **/*<data-set-name>*(*<member-name>*)**: To retrieve data from a member of a PDS or PDSE.
- **/-(*<volser>*)/*<data-set-name>***: To retrieve data from an uncataloged sequential data set.
- **/-(*<volser>*)/*<data-set-name>*(*<member-name>*)**: To retrieve data from a member of an uncataloged PDS or PDSE.

Optional Query Parameters

search=<string>
The data set is searched for the first record that contains the string, without respect to case (by default).

Optionally, insensitive=false may be specified for case sensitive matching.

This parameter may not be specified with the research= parameter.

research=<regular-expression>
The data set is searched for the first record that matches the given extended regular expression.

This parameter may not be specified with the search= parameter.

Implementation note: the regcomp() C Library function with the REG_EXTENDED flag is used.

insensitive=true|false
The default is 'true'. When 'true', searches (search and research) are case insensitive. For case sensitive searches, specify 'false'.

maxreturnsize=<integer>
This parameter may be specified only with search= or research=.

The value given is the maximum number of records to return.

The default, if not specified, is 100.

For the search and research queries, records are returned starting with the first matching record. The X-IBM-Record-Range request header may be used to specify the range of records to be searched, but it will not restrict the number of records returned (see maxreturnsize).
If no X-IBM-Record-Range request header is present, the search will begin with the first record. In all cases, an X-IBM-Record-Range=p,q response header will be returned where p is the first matching record and q is the number of records returned.
If no matching records are found, the response header X-IBM-Record-Range=0,0 will be returned.
The parameter may not be used if a request header X-IBM-Data-Type specifies any option except 'text'.

Standard headers

You can include the following standard HTTP header with this request:

If-None-Match
This header is optional; use it to specify the ETag token to be used to correlate this request with a previous request. If the data on the z/OS host has not changed since the ETag token was generated, z/OSMF returns a status of `HTTP 304 Not Modified`.

For the initial request to the resource, you can omit this header.

Note: If this header is used with very large data sets then performance may be impacted since the data set may have to be read twice by the system. This header is ignored if X-IBM-Record-Range is specified (see below). The ETag response header may be returned containing a hash string. See "X-IBM-Return-Etag" for details on whether this header will be present.

Custom headers

You can include the following custom HTTP header with this request:

X-IBM-Data-Type
This header is optional; use it to indicate whether data conversion is to be performed on the returned data, as follows:

- When set to `text`, data conversion is performed. The data transfer process converts each record from EBCDIC to the charset specified on the "Content-Type" header of the request. If no charset is specified, the default is ISO8859-1. A newline (NL) character from the response charset is inserted between logical records. For data sets with fixed-length records, trailing blanks are removed.
- When set to `binary`, no data conversion is performed. The data transfer process returns each record as-is, without translation. No delimiters are added between records. The response Content-Type is "application/octet-stream".
- When set to `record`, no data conversion is performed. Each logical record is preceded by the 4-byte big endian record length of the record that follows. This length does not include the prefix length. For example: a zero-length record is 4 bytes of zeros with nothing following.

If you omit this header, the default is `text`; the response is converted.

X-IBM-Return-Etag

This header is optional; set it to 'true' to force the response to include an "Etag" header, regardless of the size of the response data. If this header is not present or set to something other than 'true', then the default is to only send an Etag in the response for data sets smaller than a system determined length, which is at least 8MB. If X-IBM-Record-Range is present, then this header may not be specified with the value "true" and an Etag will never be returned.

If this header is enabled for very large data sets, then performance is impacted since the data set must be read twice by the system.

X-IBM-Migrated-Recall

This header is optional; use it to specify how a migrated data set is handled. By default, a migrated data set is recalled synchronously. The following values may be specified too:

wait
This is the default value. If the data set is migrated, wait for it to be recalled before processing the request.

nowait
If the data set is migrated, request it to be recalled, but do not wait.

error
If the data set is migrated, do not attempt to recall the data set.

X-IBM-Record-Range

Use this header to retrieve a range of records from a data set. You can specify this range using either of the following formats:

SSS-EEE
Where SSS identifies the start record and EEE identifies the end record to be retrieved. Both values are relative offsets (0-based).

When EEE is set to 0, records through the end of the data set are retrieved.

When SSS is omitted (i.e. -EEE), the final EEE records of the data set are retrieved.

SSS,NNN
Where SSS identifies the start record and NNN identifies the number of records to be retrieved.

NNN must be greater than zero.

Usage notes:
If X-IBM-Record-Range is specified, then an ETag header will not be returned and the If-None-Match request header is ignored.

If X-IBM-Record-Range header is present on the request, then header X-IBM-Return-Etag=true may not be specified.

If no records are found in the range specified, an exception is returned.

X-IBM-Obtain-ENQ

This header is optional; set it to one of the following values to request that a system ENQ be obtained and held after the completion of this request. If not specified, then no ENQs will be held after the completion of this request.

EXCL
a SYSDSN/Exclusive ENQ will be held on the data set

SHRW
a SYSDSN/SHR ENQ will be held on the data set, and a SPFEDIT/EXCL ENQ will be held on the data set, including the member name if this is a request for a PDS member.

A successful response will include an X-IBM-Session-Ref response header that can be added as a request header to subsequent requests to specify affinity to the TSO address space holding this ENQ.

X-IBM-Session-Ref

This header is optional; include it with the value returned from a previous X-IBM-Session-Ref response header to indicate that your request should be executed in the TSO address space that was previously reserved with a X-IBM-Obtain-ENQ request header. This address space will not be used for other requests and if not used at least once every 10 minutes it will be terminated.

The following URL request may be used to "ping" the reserved address space to keep it alive:
GET https://zosmf1.yourco.com/zosmf/restfiles/ping HTTP/1.1
X-IBM-Session-Ref: xxxxxx

The X-IBM-Obtain-ENQ and X-IBM-Session-Ref headers are mutually exclusive.

X-IBM-Release-ENQ

This header is optional; it may be specified with a value "true" to request that the ENQ held by the associated TSO address space be released.

This header must be specified along with a valid X-IBM-Session-Ref header.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. Status code `304` indicates unchanged content when a conditional get is performed (such as when using the **If-None-Match** header with an ETag from a previous response). A status code of `4`*nn* or `5`*nn* indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example request

In the following example, the GET method is used to retrieve the contents of the member SMFPRM00 in data set SYS1.PARMLIB.

```
GET /zosmf/restfiles/ds/SYS1.PARMLIB(SMFPRM00) HTTP/1.1
```

Example response

For a successful request, the HTTP response contains the following:
- Status code indicating that the request completed (status code `200`)
- ETag that you can use on subsequent requests to test for changes to the resource
- Content-Length response header that specifies the amount of data that was returned (in bytes)
- A response body that contains the resource in plain text.

A sample response header is shown in Figure 228.

```
200 OK
Etag:  B5C6454F783590AA8EC15BD88E29EA63
Content-Type:  text/plain; charset=UTF-8
Content-Language:  en-US
Content-Length:  1944
Date:  Fri, 07 Nov 2014 02:13:07 GMT
Connection: close
```

Figure 228. Example: Returned contents of the SMFPRM00 member of sys1.parmlib

A sample response body is shown in Figure 229 on page 396.

```
ACTIVE                        /*ACTIVE SMF RECORDING*/              00010000
DSNAME(SYS1.&SMFDSN1,SYS1.&SMFDSN2,    /*SMF ON 3390        */     00020000
 SYS1.&SMFDSN3)                     /*FT: SYSAQ3, TS: SYSAQ4 */     00030000
NOPROMPT                      /*PROMPT THE OPERATOR FOR OPTIONS*/   00040000
REC(PERM)                     /*TYPE 17 PERM RECORDS ONLY*/         00050000
MAXDORM(3000)                /* WRITE AN IDLE BUFFER AFTER 30 MIN*/ 00060000
MEMLIMIT(256M)               /* 256M FOR 64 BIT APPS             */ 00061005
STATUS(003000)               /* WRITE SMF STATS AFTER HALF HOUR*/   00070000
JWT(0700)                    /* INVOKE EXIT IEFUTL AFTER  7HR 00M*/ 00080002
SID(&SYSNAME),               /* SYSTEM ID FOR 3084 - SINGLE IMAGE*/ 00090000
LISTDSN                      /* LIST DATA SET STATUS AT IPL*/       00100000
INTVAL(30)                   /* INTVAL OPTION SP430 */              00110000
SYNCVAL(00)                  /* SYNCVAL OPTION SP430  */            00120000
SYS(NOTYPE(19,40,92),                                               00130001
 EXITS(IEFU83,IEFU84,IEFACTRT,IEFUJV,IEFUJI,                        00140000
          IEFUSI,IEFUTL,IEFU29),INTERVAL(010000),DETAIL)            00150000
                                                                    00160000
/* WRITE ALL RECORDS AS THE SYSTEM DEFAULT, TAKE ALL KNOWN         00170000
   EXITS, NOTE: JES EXITS CONTROLED BY JES , THERE IS NO            00180000
   DEFAULT INTERVAL RECORDS WRITTEN AND ONLY SUMMARY T32            00190000
   RECORDS AS A DEFAULT FOR TSO */                                  00200000
                                                                    00210000
SUBSYS(STC,NOTYPE(19,40,92),                                        00220001
 EXITS(IEFU29,IEFU83,IEFU84,IEFUTL),                                00230000
 INTERVAL(SMF,SYNC),DETAIL)           /*SP430*/                    00240000
                                                                    00250000
/* WRITE ALL RECORDS AS BY  SYSTEM DEFAULT, TAKE ONLY THREE        00260000
   EXITS, NOTE: IEFU29 EXECUTES IN THE MASTER ASID WHICH IS A       00270000
   STC ADDRESS SPACE SO IEFU29 MUST BE ON FOR STC. USE ALL OTHER    00280000
   SYS PARAMETERS AS A DEFAULT  */                                  00290000
```

Figure 229. Example: Returned contents of the SMFPRM00 member of sys1.parmlib

Example request

In the following example, the GET method is used to retrieve the contents of a sequential data set.

```
GET /zosmf/restfiles/ds/JIAHJ.REST.SRVMP HTTP/1.1
```

Example response

A sample response body is shown in Figure 230 on page 397.

Response

```
200 OK
X-Powered-By:  Servlet/3.0
Content-Type:  text/plain; charset=UTF-8
Content-Length:  2131
Etag: 47029CDDCD91E2887E1FAAD6FCD75ECB
Content-Language:  en-US
Date:  Wed, 25 Nov 2015 02:27:15 GMT
```

Response Body

```
//JH2FPROC EXEC PGM=IKJEFT01,DYNAMNBR=200
//*************************************************************/
//* TSO LOGON PROC FOR Z/OS DATA SET AND FILE REST INTERFACE  */
//*                                */
//* PROPRIETARY STATEMENT:                    */
//*                                */
//*     LICENSED MATERIALS - PROPERTY OF IBM        */
//*     5610-A01                       */
//*     COPYRIGHT IBM CORP. 2014            */
//*     STATUS = HSMA210                  */
//*************************************************************/
//CEEOPTS  DD *
DYNDUMP(*USERID.PRIVATE)
//SYSEXEC  DDDISP=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=JIAHJ.REST.LMOD
//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 DISP=SHR,DSN=JIAHJ.REST.PARMLIB(IZUSRVMP)
//SYSOUT   DD SYSOUT=H
//CEEDUMP  DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//
```

Figure 230. Example: Retrieve the contents of a sequential data set

Write data to a z/OS data set or member

You can use this operation to write data to an existing sequential data set, or a member of a partitioned data set (PDS or PDSE). To write to an uncataloged data set, include a volume serial on the request.

HTTP method and URI path

```
PUT /zosmf/restfiles/ds/[-(<volser>)/]<dataset-name>[(<member-name>)]
```

---

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- ***-(<volser>)*** represents a volume serial. For an uncataloged data set, include this parameter to identify the volume to be searched for data sets or members that match the specified *<data-set-name>* or *<member-name>*. The length of the volume serial cannot exceed six characters. You cannot use wildcard characters for this parameter. Indirect volume serials are not supported.
- ***<dataset-name>*** identifies the data set to which to write. This parameter is required and must consist of a fully qualified data set name. The length of the data set name that you specify on the request cannot exceed 44 characters.
- ***<member-name>*** identifies the name of the PDS or PDSE member to which to write. Include this parameter for a PDS or PDSE member write request.

  If the member does not exist, it is created. If the data set name identifies a base name of a Generation Data Group (GDG), then member may refer to relative data sets, for example: (0), (+1), (-1)

Based on the object to which you want to write, you can specify one of the following parameter combinations:

- **/*<data-set-name>***: To write to a sequential data set.
- **/*<data-set-name>*(*<member-name>*)**: To write to a member of a PDS or PDSE.
- **/-(*<volser>*)/*<data-set-name>***: To write to an uncataloged sequential data set.
- **/-(*<volser>*)/*<data-set-name>*(*<member-name>*)**: To write to a member of an uncataloged PDS or PDSE.

Request body

The data to write to the target data set. The data is interpreted according to the content-type as one of binary, text, record or 'diff -e' format according a combination of the "Content-Type" and the value of the X-IBM-Data-Type custom header, if present.

Standard headers

You can include the following standard HTTP header with this request:

`If-Match`

This header is optional; use it to specify the ETag to be used for correlating this request with a previous request on the same resource. If the resource has not changed since the ETag token was generated, the data is written to the target data set or member. Otherwise, if the resource has been modified, the request is failed with status code `HTTP 412`.

If you omit this header, the data is always written, regardless of whether the resource is changed.

Custom headers

You can use the following custom HTTP header with this request:

X-IBM-Data-Type

This header is optional; use it to indicate whether data conversion is to be performed on the request body.

text

When set to `text`, data conversion is performed. The data transfer process converts each record from the charset specified on the "Content-Type" header of the request. If no charset is specified, the default is ISO8859-1. Each line of data, delimited by a Line Feed in the request charset, is converted to EBCDIC and written as a record to the data set or member. (The line feed character is removed from the data, and the data is padded with the space character to the end of the record if it is a fixed record size data set. For variable record size data sets, the record is written without padding.) If the record size of the data set is smaller than any line of text, an HTTP 400 is returned with a JSON error document indicating that not all data was written.

Note: When set to 'text' and "Content-Type" is "application/x-ibm-diff-e", the input consists of commands in the same format as produced by the z/OS UNIX 'diff -e' command. These commands are used to add, replace and delete lines in the target data set. The following commands are supported:
a
c
d
s/.//

Each command may be optionally preceded by a line or line range, as allowed by the z/OS UNIX 'ed' command. If an error is detected while processing a command,status code 500 is returned with an exception.

binary

When set to `binary`, no data conversion is performed. The data is written to the data set without respect to record boundaries. All records will be written at their maximum record length and for fixed length record data sets, the last record will be padded with nulls if required.

record

When set to `record`, no data conversion is performed. Each logical record is preceded by the 4-byte big endian record length of the record that follows. This length does not include the prefix length. For example: a zero-length record would be 4 bytes of zeros with nothing following.

If you omit this header, the default is `text`; the request body is converted.

X-IBM-Migrated-Recall

This header is optional; use it to specify how a migrated data set is handled. By default, a migrated data set is recalled synchronously. The following values may be specified too:

wait

This is the default value. If the data set is migrated, wait for it to be recalled before processing the request.

nowait

If the data set is migrated, request it to be recalled, but do not wait.

error

If the data set is migrated, do not attempt to recall the data set.

X-IBM-Obtain-ENQ

This header is optional; set it to one of the following values to request that a system ENQ be obtained and held after the completion of this request. If not specified, then no ENQs will be held after the completion of this request.

EXCL
a SYSDSN/Exclusive ENQ will be held on the data set

SHRW
a SYSDSN/SHR ENQ will be held on the data set, and a SPFEDIT/EXCL ENQ will be held on the data set, including the member name if this is a request for a PDS member.

A successful response will include an X-IBM-Session-Ref response header that can be added as a request header to subsequent requests to specify affinity to the TSO address space holding this ENQ.

X-IBM-Session-Ref

This header is optional; include it with the value returned from a previous X-IBM-Session-Ref response header to indicate that your request should be executed in the TSO address space that was previously reserved with a X-IBM-Obtain-ENQ request header. This address space will not be used for other requests and if not used at least once every 10 minutes it will be terminated.

The following URL request may be used to "ping" the reserved address space to keep it alive:
GET https://zosmf1.yourco.com/zosmf/restfiles/ping HTTP/1.1
X-IBM-Session-Ref: xxxxxx

The X-IBM-Obtain-ENQ and X-IBM-Session-Ref headers are mutually exclusive.

X-IBM-Release-ENQ

This header is optional; it may be specified with a value "true" to request that the ENQ held by the associated TSO address space be released.

This header must be specified along with a valid X-IBM-Session-Ref header.

Required authorizations

See "Required authorizations" on page 384.

Request content

Your request must supply the data set content. For an example, see "Example request."

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `204` indicates success. Status code `201` indicates success if a new PDS member was created. Status code `412` indicates that the document does not match the supplied ETag token on the If-Match header as described above. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example request

Suppose that you want to update the contents of the SMFPRM00 member of SYS1.PARMLIB using a PUT request. Figure 231 on page 401 shows an example of the request header that you might use.

```
PUT /zosmf/restfiles/ds/SYS1.PARMLIB(SMFPRM00)
If-Match: B5C6454F783590AA8EC15BD88E29EA63
Content-Type: text/plain; charset=UTF-8
```

Figure 231. Example: Request header for a write request to the SMFPRM00 member of sys1.parmlib

In Figure 231, notice that the optional header If-Match is included. This header is specified with an ETag that was obtained from a previous read request on the parmlib member. Using an ETag in this manner allows for conditional processing; the new member contents are written only when the member has not been modified on the host system since the ETag was generated. If the member was modified, for example, by another user or process, this request is failed with HTTP status code 412.

A sample request body is shown in Figure 232. The body contains the new contents of the member.

```
/*****************************************************/
/* THIS PARMLIB MEMBER CONTAINS CONFIGURATION FOR SMF */
/*****************************************************/
 ACTIVE                        /*ACTIVE SMF RECORDING*/              00010000
 DSNAME(SYS1.&SMFDSN1,SYS1.&SMFDSN2,    /*SMF ON 3390        */     00020000
  SYS1.&SMFDSN3)                    /*FT: SYSAQ3, TS: SYSAQ4 */     00030000
 NOPROMPT                     /*PROMPT THE OPERATOR FOR OPTIONS*/   00040000
 REC(PERM)                    /*TYPE 17 PERM RECORDS ONLY*/         00050000
 MAXDORM(3000)               /* WRITE AN IDLE BUFFER AFTER 30 MIN*/ 00060000
 MEMLIMIT(256M)              /* 256M FOR 64 BIT APPS             */ 00061005
 STATUS(003000)              /* WRITE SMF STATS AFTER HALF HOUR*/   00070000
 JWT(0700)                   /* INVOKE EXIT IEFUTL AFTER  7HR 00M*/ 00080002
 SID(&SYSNAME),              /* SYSTEM ID FOR 3084 - SINGLE IMAGE*/ 00090000
 LISTDSN                     /* LIST DATA SET STATUS AT IPL*/       00100000
 INTVAL(30)                  /* INTVAL OPTION SP430 */              00110000
 SYNCVAL(00)                 /* SYNCVAL OPTION SP430  */            00120000
 SYS(NOTYPE(19,40,92),                                             00130001
  EXITS(IEFU83,IEFU84,IEFACTRT,IEFUJV,IEFUJI,                      00140000
       IEFUSI,IEFUTL,IEFU29),INTERVAL(010000),DETAIL)              00150000
                                                                   00160000
 /* WRITE ALL RECORDS AS THE SYSTEM DEFAULT, TAKE ALL KNOWN        00170000
    EXITS, NOTE: JES EXITS CONTROLED BY JES , THERE IS NO          00180000
    DEFAULT INTERVAL RECORDS WRITTEN AND ONLY SUMMARY T32          00190000
    RECORDS AS A DEFAULT FOR TSO */                                00200000
                                                                   00210000
 SUBSYS(STC,NOTYPE(19,40,92),                                      00220001
  EXITS(IEFU29,IEFU83,IEFU84,IEFUTL),                              00230000
  INTERVAL(SMF,SYNC),DETAIL)           /*SP430*/                   00240000
                                                                   00250000
 /* WRITE ALL RECORDS AS BY  SYSTEM DEFAULT, TAKE ONLY THREE       00260000
    EXITS, NOTE: IEFU29 EXECUTES IN THE MASTER ASID WHICH IS A     00270000
    STC ADDRESS SPACE SO IEFU29 MUST BE ON FOR STC. USE ALL OTHER  00280000
    SYS PARAMETERS AS A DEFAULT  */                                00290000
```

Figure 232. Example: Request body for a write request to the SMFPRM00 member of sys1.parmlib

Example response

For a successful request, the HTTP response contains the following:
- Status code indicating that the request completed (status code 204)
- ETag that you can use on subsequent requests to test for changes to the resource

```
204 No Content
Etag:  DE2BE8B8485EB8F1E28D3716DFFE0680
Content-Type:  application/json; charset=UTF-8
Content-Language:  en-US
Date:  Fri, 07 Nov 2014 02:31:39 GMT
```

Example request

The PUT method is used to write the contents of a sequential data set.

```
PUT /zosmf/restfiles/ds/JIAHJ.REST.SRVMP HTTP/1.1
If-Match: B5C6454F783590AA8EC15BD88E29EA42
Content-Type: text/plain; charset=UTF-8
```

Example response

A sample response is shown in Contents of a sequential data set.

Response:

```
204 No Content
X-Powered-By:  Servlet/3.0
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Etag: 39E89731CE27214AE2FE0BBD9200DC26
Content-Language:  en-US
Date:  Wed, 25 Nov 2015 03:10:12 GMT
```

Figure 233. Example: Contents of a sequential data set

Create a sequential and partitioned data set

You can use this operation to create sequential and partitioned data sets on a z/OS system.

HTTP method and URI path

```
POST /zosmf/restfiles/ds/<dataset-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- **<dataset-name>** is the name of a z/OS data set that you are going to create.

Request Body

The request body to create a sequential and partitioned data set is shown in Request body to create a sequential and partitioned data set .

Table 218. Request body to create a sequential and partitioned data set

Field	Type	Description
volser	String	Volume.
unit	String	Device type.
dsorg	String	Data set organization.
alcunit	String	Unit of space allocation.
primary	Integer	Primary space allocation.
secondary	Integer	Secondary space allocation.
dirblk	Integer	Number of directory blocks.
avgblk	Integer	Average block.
recfm	String	Record format.
blksize	Integer	Block size.
lrecl	Integer	Record length.
storeclass	String	Storage class.
mgntclass	String	Management class.
dataclass	String	Data class.

Standard headers

None.

Custom headers

None.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code 201 indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 384.

For a successful creating request, 201 Created with no content will be returned. .

Example request

In the following example, the POST method is used to create a sequential data set.

```
POST /zosmf/restfiles/ds/JIAHJ.REST.TEST.NEWDS HTTP/1.1

Request body:
{"volser":"zmf046","unit":"3390","dsorg":"PS","alcunit":"TRK","primary":10,
"secondary":5,"avgblk":500,"recfm":"FB","blksize":400,"lrecl":80}
```

Example response

A sample response is shown in Example: Create a data set.

```
201 Created
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 16 Sep 2015 10:54:21 GMT
```

Figure 234. Example: Create a data set

Example request

The POST method is used to create a partitioned data set.

```
POST /zosmf/restfiles/ds/JIAHJ.REST.TEST.NEWDS02 HTTP/1.1

Request Body
{"volser":"zmf046","unit":"3390","dsorg":"PO","alcunit":"TRK","primary":10,
"secondary":5,"dirblk":10,"avgblk":500,"recfm":"FB","blksize":400,"lrecl":80}
```

Example response

A sample response is shown in Example: Create data set.

```
201 Created
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 16 Sep 2015 11:14:13 GMT
```

Figure 235. Example: Create data set.

Delete a sequential and partitioned data set

You can use this operation to delete sequential and partitioned data sets on a z/OS system.

HTTP method and URI path

```
DELETE /zosmf/restfiles/ds/<dataset-name>
DELETE /zosmf/restfiles/ds/-(<volume>)/<dataset-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- **<dataset-name>** is the name of a z/OS data set, that you are going to delete.
- **<volume>** is where the data set is resided, when the data set is uncatalogued.

Request Body

None.

Standard headers

None.

Custom headers

None.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `204` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

If the request is successfully executed, status code 204 indicates success and no content is returned.

Example request

In the following example, the DELETE method is used to delete a data set.

```
DELETE /zosmf/restfiles/ds/JIAHJ.REST.TEST.DATASET HTTP/1.1
```

Example response

A sample response is shown in Figure 236.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 16 Sep 2015 12:08:38 GMT
```

Figure 236. Example: Delete a data set

Example request

The DELETE method is used to delete an uncatalogued data set.

```
DELETE /zosmf/restfiles/ds/-(ZMF046)/JIAHJ.REST.TEST.DATASET2 HTTP/1.1
```

Example response

A sample response is shown in Delete uncatalogued data set.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 16 Sep 2015 12:10:22 GMT
```

Figure 237. Example: Delete uncatalogued data set.

Delete a partitioned data set member

You can use this operation to delete a member of a PDS or PDSE.

HTTP method and URI path

```
DELETE /zosmf/restfiles/ds/<dataset-name>(<member-name>)
DELETE /zosmf/restfiles/ds/-(volume)/<dataset-name>(<member-name>)
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a data set request
- **<dataset-name>** is the name of a z/OS data set that contains a member you are going to delete.
- **<member-name>** is the name of the partitioned data set member, that you are going to delete.
- **<volume>** is where the data set resides, when the data set is uncatalogued.

Request Body

None.

Standard headers

None.

Custom headers

None.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `204` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

If the request is successfully executed, status code 204 indicates success and no content is returned.

Example request

In the following example, the DELETE method is used to delete a member of a cataloged partitioned data set.

```
DELETE zosmf/restfiles/ds/JIAHJ.REST.TEST.PDS(MEMBER01) HTTP/1.1
```

Example response

A sample response is shown in Delete a member of a cataloged partitioned data set.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Tue, 15 Sep 2015 10:36:14 GMT
```

Figure 238. Example: Delete a member of a cataloged partitioned data set

Example request

The DELETE method is used to delete a member of an uncataloged partitioned data set.

```
DELETE zosmf/restfiles/ds/-(ZMF046)/JIAHJ.REST.TEST.PDS.UNCAT(MEMBER01) HTTP/1.1
```

Example response

A sample response is shown in Delete a member of an uncataloged partitioned data set.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Tue, 15 Sep 2015 11:37:12 GMT
```

Figure 239. Example: Delete a member of an uncataloged partitioned data set

z/OS Data set and member utilities

You can use the Data set and member utilities to operate on data sets or members. Operations include: rename data set, rename member, copy data set and copy member, migrate data set, recall a migrated data set, delete a backup version of a data set.

HTTP method and URI path

```
PUT /zosmf/restfiles/ds/<to-data-set-name>
PUT /zosmf/restfiles/ds/<to-data-set-name>(<member-name>)
```

Figure 240. 'rename' request

```
PUT /zosmf/restfiles/ds/<to-data-set-name>
PUT /zosmf/restfiles/ds/<to-data-set-name>(<member-name>)
PUT /zosmf/restfiles/ds/-(<to-volser>)/<to-data-set-name>
PUT /zosmf/restfiles/ds/-(<to-volser>)/<to-data-set-name>(<member-name>)
```

Figure 241. 'copy' request

```
PUT /zosmf/restfiles/ds/<to-data-set-name>
```

Figure 242. "hmigrate' , 'hrecall', or 'hdelete' request

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ds** indicates a Dataset request
- ***-(<volser>)*** represents a volume serial. For an uncataloged data set, include this parameter to identify the volume to be searched for data sets or members that match the specified *<data-set-name>* or *<member-name>*. The length of the volume serial cannot exceed six characters. You cannot use wildcard characters for this parameter. Indirect volume serials are not supported.
- ***<to-data-set-name>*** identifies the target data set name. This parameter is required and must consist of a fully qualified data set name. The length of the data set name that you specify on the request cannot exceed 44 characters.
- ***<member-name>*** identifies the target PDS or PDSE member name.

Headers

X-IBM-BPXK-AUTOCVT

This header is optional; use it to indicate how file auto conversion is handled when using the copy operation to copy text mode data sets to POSIX files, if you omit this header, the system default is taken.

'on' or 'all'

The target file is a candidate for automatic conversion if its TXTFLAG is tagged TEXT and the source data set is type TEXT.

'off'

The target file is not a candidate for automatic conversion

X-IBM-Migrated-Recall

This header is optional; use it to specify how a migrated data set is handled. By default, a migrated data set is recalled synchronously. The following values may be specified too:

wait
This is the default value. If the data set is migrated, wait for it to be recalled before processing the request.

nowait
If the data set is migrated, request it to be recalled, but do not wait.

error
If the data set is migrated, do not attempt to recall the data set.

The header, Content-Type: application/json; charset={charset-name}, must be specified as well.

Request body

A JSON request document (content-type=application/json, character-encoding=UTF-8) must be supplied in one of the following forms:

Table 219. Request

Function	Property	Description	Required
hmigrate	wait:true | false	If true then the function waits for completion of the request. If false the request is queued.	
hrecall	wait:true | false	If true then the function waits for completion of the request. If false the request is queued.	
hdelete	wait:true | false	If true then the function waits for completion of the request. If false the request is queued.	
	purge:true | false	If true then the function uses the PURGE=YES on ARCHDEL request. If false the function uses the PURGE=NO on ARCHDEL request.	
rename	request	Indicates the function name.	Yes
	from-dataset	The data set to rename.	Yes
		dsn The source data set name. This is required	Yes
		member If renaming a member this is the old member name. This is not required.	No
	enq	enq for the "to" data set is only allowed for renaming members. Values may be SHRW or EXCLU. SHRW is the default or PDS members, EXCLU otherwise.	No

Table 219. Request (continued)

Function	Property	Description	Required
copy	request	Indicates the function copy.	Yes
	from-file	The file to copy.	You must choose either from-file or from-dataset.
		filename The absolute source filename. This is required.	
		type One of "binary | executable | text". Default is text. This is not required.	
	from-dataset	The dataset to copy.	
		dsn The source data set. This is required.	
		member Used to specify a member; "*" means all members. This is not required.	
		volser May be specified if dsn is not cataloged. This is not required.	
		alias:true|false if true, aliases are copied along with main member;if false(default), alias relationships are not maintained. This is not required.	
	enq	Only applicable when from-dataset specified. With from-file, an error is reported (see note below).This is the enqueue type for the "to" data set. Allowed values are: SHR, SHRW, EXCLU;SHRW is the default for PDS, EXCLU for sequential. The source data set is always enqueued via SHR. **Note:** When from-file is specified, the target dsn is opened with DISP=OLD (EXCLU) with one exception: if the target is a PDS and the from-file/type is text, the target PDS is enqueued SHRW. This is not required.	No
	replace:true|false	Applicable with from-dataset. When from-file specified,ignored unless from-file/type=text. if true, members in the target dataset are replaced. if false(default), like named members are not copied and an error is returned.	No

Note: The "to" data set must be a PDS if from-dataset/member is '*' or a <member-name> is specified on the URL. When from-dataset/member is a single member name and the member name is NOT specified on the URL, the 'to' data set is expected to be sequential.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200 OK` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example

Refer to Figure 243 for an example of renaming a data set.

```
Request:
PUT https://zosmf1.yourco.com/zosmf/restfiles/ds/MY.NEW.DSN HTTP/1.1
Content-Type: application/json; charset=UTF-8

{"request":"rename", "from-dataset":{"dsn":"MY.OLD.DSN"}}
```

Figure 243. Example: Rename MY.OLD.DSN to MY.NEW.DSN

Example

Refer to Figure 244 for an example of copying a PDS member.

```
Request:
PUT https://zosmf1.yourco.com/zosmf/restfiles/ds/MY.NEW.DSN(MYMEM2) HTTP/1.1
Content-Type: application/json; charset=UTF-8

{"request":"copy", "from-dataset":{"dsn":"MY.OLD.DSN", "member":"MYMEM1"}, "replace":true }
```

Figure 244. Example: copy member MYMEM1 from MY.OLD.DSN to MY.NEW.DSN(MYMEM2)

Access Method Services Interface

You can use the Access Method Service (IDCAMS) to provide a REST/JSON interface to IDCAMS. You can use this operation to create a new zFS filesystem.

HTTP method and URI path

```
PUT /zosmf/restfiles/ams
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/ams** indicates a request for Access Method Services (IDCAMS) services.

Input Content

Input content in a json document:

Table 220. Input Content

Property	Description	Required
input	one or more input lines <= 255 in length	Yes
JSONversion:1	JSON Version	No

Note: The size of all input lines plus the number of input lines must be <= 8K.

Response Body

If the request is successfully executed, will return 200 status code (IDCAMS RC<=4). In all cases an application/json document will be returned:

Table 221. Response

Property	Description	Required
rc	Return code from IDCAMS.	Yes
output	One or more input lines <= 255 in length.	Yes
JSONversion	JSON Version.	No

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200 OK` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see “Error handling” on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example

Refer to Figure 245 for an example of IDCAMS Access Methods Services.

```
request:
PUT https://zosmf1.yourco.com/zosmf/restfiles/ams  HTTP/1.1
 Content-Type: application/json
 Content-Length: nn
 {
  "input":[
  "DEFINE CLUSTER(NAME (EXAMPL1.KSDS) VOLUMES(VSER05)) -",
  "DATA  (KILOBYTES (50 5))"],
  "JSONversion":1
 }
```

Figure 245. IDCAMS Access Methods Services

List the files and directories of a UNIX file path

You can use this operation to list the files and directories in a UNIX file path on a z/OS system.

HTTP method and URI path

```
GET /zosmf/restfiles/fs?path=<filepath-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/fs** identifies a UNIX file path request
- **?path=<*filepath-name*>** is a query parameter that specifies the directory containing the files and directories to be listed.

Standard headers

None.

Custom headers

You can include the following custom HTTP header with this request:

X-IBM-Max-Items
: This header value specifies the maximum number of items to return. To request that all items be returned, set this header to 0. If you omit this header, or specify an incorrect value, up to 1000 items are returned by default.

X-IBM-Lstat
: If the value of this header is "true", an lstat() is performed on the path rather than stat() and a list containing one item is returned with the lstat results. For more information regarding lstat() and stat(), please refer to the *z/OS XL C/C++ Runtime Library Reference* .

Query parameters

Specify the following query parameter on this request:

path
: This parameter identifies the UNIX directory that contains the files and directories to be listed. This parameter is required and can consist of one or more directories in the hierarchical file system structure, or a fully qualified file name. A fully qualified file name, which consists of the name of each directory in the path to a file plus the file name itself, can be up to 1023 bytes long. You cannot use wildcard characters for this parameter.

  The following list contains sample file path names:

  ```
  /
  /bin
  /usr/lib/libSM.a
  ```

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For a successful request, the HTTP response includes an array of matching of UNIX files and directories, each as a JSON list document. For the contents, see "File list document" on page 451.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example request

In the following example, the GET method is used to list the files and directories in the UNIX path `/usr`:

```
GET /zosmf/restfiles/fs?path=/usr HTTP/1.1
```

Example response

A sample response is shown in Figure 246.

Response

```
200 OK
X-Powered-By:  Servlet/3.0
Content-Type:  application/json; charset=UTF-8
Content-Length:  896
Content-Language:  en-US
Date:  Tue, 24 Nov 2015 06:12:16 GMT
```

Response Body

```
{"items":[
{"name":".", "mode":"drwxrwxrwx", "size":8192, "uid":0, "user":"WSADMIN", "gid":1,
"group":"OMVSGRP", "mtime":"2015-11-24T02:12:04"},
{"name":"..", "mode":"drwxr-xr-x", "size":8192, "uid":0, "user":"WSADMIN", "gid":1,
"group":"OMVSGRP", "mtime":"2015-09-15T02:38:29"},
{"name":".profile", "mode":"-rwxrwxrwx", "size":849, "uid":0, "user":"WSADMIN", "gid":1,
"group":"OMVSGRP", "mtime":"2013-02-13T12:08:29"},
{"name":".sh_history", "mode":"-rw-------", "size":4662, "uid":0, "user":"WSADMIN", "gid":1,
"group":"OMVSGRP", "mtime":"2013-06-06T18:09:28"},
{"name":"myFile.txt", "mode":"-rw-r--r--", "size":20, "uid":0, "user":"WSADMIN", "gid":1,
"group":"OMVSGRP", "mtime":"2015-11-24T02:12:04"},
{"name":"profile.add", "mode":"-rwxrwxrwx", "size":888, "uid":0, "user":"WSADMIN", "gid":1,
"group":"OMVSGRP", "mtime":"2013-05-07T11:23:08"}
],"returnedRows":6,"totalRows":6,"JSONversion":1}
```

Figure 246. Example: Returned list of UNIX files and directories in path `/usr`

Example request

In the following example, the GET method is used to list a Unix file

```
GET /zosmf/restfiles/fs?path=/u/ibmuser/myFile.txt HTTP/1.1
```

Example response

A sample response is shown in Example: Returned list of UNIX files.

Response

```
200 OK
X-Powered-By:  Servlet/3.0
Content-Type:  application/json; charset=UTF-8
Content-Length:  210
Content-Language:  en-US
Date:  Tue, 24 Nov 2015 09:16:49 GMT
```

Response Body

```
{"items":[
{"name":"/u/ibmuser/myFile.txt", "mode":"-rw-r--r--", "size":20, "uid":0, "user":"WSADMIN",
"gid":1, "group":"OMVSGRP", "mtime":"2015-11-24T02:12:04"}
],"totalRows":1,"returnedRows":1},"JSONversion":1}
```

Figure 247. Example: Returned list of UNIX files

Retrieve the contents of a z/OS UNIX file

You can use this operation to retrieve the contents of a z/OS UNIX System Services file.

HTTP method and URI path

```
GET /zosmf/restfiles/fs/<filepath-name>
```

---

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/fs** indicates a UNIX file request
- ***<filepath-name>*** identifies the UNIX file to be read. This parameter is required and must consist of a fully qualified path and file name.

Optional Query Parameters

search=<string>
The data set is searched for the first record that contains the string, without respect to case (by default).

Optionally, insensitive=false may be specified for case sensitive matching.

This parameter may not be specified with the research= parameter.

research=<regular-expression>
The data set is searched for the first record that matches the given extended regular expression.

This parameter may not be specified with the search= parameter.

Implementation note: the regcomp() C Library function with the REG_EXTENDED flag is used.

insensitive=true|false
The default is 'true'. When 'true', searches (search and research) are case insensitive. For case sensitive searches, specify 'false'.

maxreturnsize=<integer>
This parameter may be specified only with search= or research=.

The value given is the maximum number of records to return.

The default, if not specified, is 100.

For the search and research queries, records are returned starting with the first matching record. The X-IBM-Record-Range request header may be used to specify the range of records to be searched, but it will not restrict the number of records returned (see maxreturnsize).
If no X-IBM-Record-Range request header is present, the search will begin with the first record. In all cases, an X-IBM-Record-Range=p,q response header will be returned where p is the first matching record and q is the number of records returned.
If no matching records are found, the response header X-IBM-Record-Range=0,0 will be returned.
The parameter may not be used if a request header X-IBM-Data-Type specifies any option except 'text'.

Standard headers

You can include the following standard HTTP header with this request:

If-None-Match
This header is optional; use it to specify the ETag token to be used to correlate this request with a previous request. If the data on the z/OS host has not changed since the ETag token was generated, z/OSMF returns a status of `HTTP 304 Not Modified`.

For an initial request to the resource, you can omit this header.

Range

This header is optional; use this header to retrieve a range of bytes from a file. This header is supported only when X-IBM-Data-Type=binary. Specify this range using the following:
bytes=first-byte-pos "-" last-byte-pos

The first-byte-pos value is the byte-offset of the first byte in a range. The last-byte-pos value is the byte-offset of the last byte in the range. The byte positions specified are inclusive. Byte offsets start at zero. When last-byte-pos is not specified or is zero, the range extends to the end of the file. When the first-byte-pos is not specified, a tail range is returned. Comma separated ranges are not supported. For an initial request to the resource, you can omit this header.

Usage notes: If the range cannot be satisfied, i.e. zero bytes are returned, then a status code of 416 is set.

Examples (assuming a file with 10000 bytes):
bytes=0-499 retrieves the first 500 bytes
bytes=500-999 retrieves the second 500 bytes
bytes=500- retrieves the final 9500 bytes
bytes=-500 retrieves the final 500 bytes

X-IBM-Record-Range

Use this header to retrieve a range of records (lines delimited by '\n') from a file. You can specify this range using either of the following formats:

SSS-EEE
Where SSS identifies the start record and EEE identifies the end record to be retrieved. Both values are relative offsets (0-based). When EEE is set to 0, records through the end of the file are retrieved. When SSS is omitted (i.e. -EEE), the final EEE records of the file are retrieved.

SSS,NNN
Where SSS identifies the start record and NNN identifies the number of records to be retrieved.

Usage notes: If X-IBM-Record-Range is specified with Range an error is reported. If zero bytes are returned due to the range specified, status code 500 is returned.

Custom headers

You can include the following custom HTTP header with this request:

X-IBM-Data-Type

This header is optional; use it to indicate whether data conversion is to be performed on the returned data, as follows:

- When set to `text`, data conversion is performed. The data transfer process converts each record from EBCDIC to the charset specified on the "Content-Type" header of the request. If no charset is specified, the default is ISO8859-1. A newline (NL) character from the response charset is inserted between logical records. For data sets with fixed-length records, trailing blanks are removed.
- When set to `binary`, no data conversion is performed. The data transfer process returns each line of data as-is, without translation.

If you omit this header, the default is `text`; the response is converted.

Query parameters

None.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. Status code `304` indicates an unchanged file when a conditional get is performed (such as when using the **`If-None-Match`** header with an ETag from a previous response). Status code `206` indicates that a part of the file has been returned as a result of a Range header on the request. Accompanying this status code will be a Content-Range header in the form sss-eee/nnnnnn where sss-eee is the byte range that was actually returned and nnnnnn is the length of the file. This status is returned only for the standard range header, not the custom X-IBM-Record-Range header. Status code `416` indicates that zero bytes have been returned due to the Range header on the request. This status is returned only for the standard range header, not the custom X-IBM-Record-Range header. A status code of `4`*`nn`* or `5`*`nn`* indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example request

In the following example, the GET method is used to retrieve the contents of the file inetd.conf in the `/etc` directory.

```
GET /zosmf/restfiles/fs/etc/inetd.conf HTTP/1.1
```

Example response

For a successful request, the HTTP response contains the following:

- Status code indicating that the request completed (status code `200`)
- ETag that you can use on subsequent requests to test for changes to the resource
- Content-Length response header that specifies the amount of data that was returned (in bytes)
- A response body that contains the resource in plain text.

```
200 OK
X-Powered-By:  Servlet/3.0
Content-Type:  text/plain; charset=UTF-8
Content-Length:  2673
Etag: AEA05EC01C7922ADD5103EBD95FFCC58
Content-Language:  en-US
Date:  Wed, 25 Nov 2015 03:07:10 GMT
```

A sample response body is shown in Figure 248 on page 422.

```
###                                                                          00000100
# Used to replace /etc/inetd.conf on 2nd level z/OS system                   00000101
# so we can telnet/rlogin directly to OMVS using PuTTY consoles.             00000102
###                                                                          00000110
# Internet server configuration database                                     00000200
#                                                                            00000300
# (C) COPYRIGHT International Business Machines Corp. 1985, 2001             00000400
# All Rights Reserved                                                        00000500
# Licensed Materials - Property of IBM                                       00000600
#                                                                            00000700
# US Government Users Restricted Rights - Use, duplication or                00000800
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.          00000900
#                                                                            00001000
# /etc/inetd.conf                                                            00001100
#                                                                            00001200
#          Internet server configuration database                            00001300
#                                                                            00001400
#  $01=PYQ0049,  HOT7705, 010130, PDJP: Correct paths and remove             00001500
#        unsupported services (FIN APAR OW45915                              00001600
#                                                                            00001700
# Services can be added and deleted by deleting or inserting a               00001800
# comment character (ie. #) at the beginning of a line                       00001900
#                                                                            00002000
#====================================================================        00002100
# service | socket | protocol | wait/ | user | server  | server program    00002200
# name    | type   |          | nowait|      | program |   arguments       00002300
#====================================================================        00002400
#                                                                            00002500
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l          00002600
#sh      stream tcp nowait OMVSKERN /usr/sbin/sshd sshd -i
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV               00002700
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m            00002800
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV           00002900
```

Figure 248. Example: Response body for a GET request to the UNIX file /etc/inetd.conf

Write data to a z/OS UNIX file

You can use this operation to write data to an existing z/OS UNIX System Services file.

HTTP method and URI path

```
PUT /zosmf/restfiles/fs/<filepath-name>
```

---

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/fs** indicates a UNIX file request
- ***<filepath-name>*** identifies the UNIX file to which to write. This parameter is required and must consist of a fully qualified path and file name.

Request body

The data to write to the target data set. The data is interpreted according to the content-type as one of binary, text, record or 'diff -e' format according a combination of the "Content-Type" and the value of the X-IBM-Data-Type custom header, if present.

Standard headers

You can include the following standard HTTP header with this request:

If-Match
This header is optional; use it to specify the ETag to be used for correlating this request with a previous request on the same UNIX file. If the file has not changed since the ETag was generated, the request is processed. Otherwise, if the file has been modified, the request is failed with status code `HTTP 412`.

If you omit this header, the data is always written, regardless of whether the file is changed.

Custom headers

You can include the following custom HTTP header with this request:

X-IBM-Data-Type
This header is optional; use it to indicate whether data conversion is to be performed on the data to be written, as follows:

text
When set to `text`, data conversion is performed. The data transfer process converts each byte from the charset specified on the "Content-Type" header of the request. If no charset is specified, the default is ISO8859-1. Line Feed characters are left intact. This is the default value

Note: When set to 'text' and "Content-Type" is "application/x-ibm-diff-e", the input consists of commands in the same format as produced by the z/OS UNIX 'diff -e' command. These commands are used to add, replace and delete lines in the target data set. The following commands are supported:
a
c
d
s/.//

Each command may be optionally preceded by a line or line range, as allowed by the z/OS UNIX 'ed' command. If an error is detected while processing a command,status code 500 is returned with an exception.

binary
When set to `binary`, no conversion is performed.

If you omit this header, the default is `text`; the data is converted.

Query parameters

None.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `204` indicates success. A status code of `4`*nn* or `5`*nn* indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example request

In the following example, the PUT method is used to replace the UNIX file `/etc/inetd.conf`.

```
PUT /zosmf/restfiles/fs/etc/inetd.conf HTTP/1.1
If-Match: F4A5A479E78AFD4CFF7DF13937AB82AE
Content-Type: text/plain; charset=UTF-8
```

A sample request body is shown in Figure 249 on page 425.

```
###
# Internet server configuration database
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 2001
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# /etc/inetd.conf
#
#          Internet server configuration database
#
#  $01=PYQ0049,  HOT7705, 010130, PDJP: Correct paths and remove
#        unsupported services (FIN APAR OW45915
#
# Services can be added and deleted by deleting or inserting a
# comment character (ie. #) at the beginning of a line
#
#========================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#========================================================================
#
# Following line uncommented by USSSETUP job: 2013/04/24 15:04:00
otelnet   stream tcp nowait IBMUSER  /usr/sbin/otelnetd otelnetd -l
# Following line uncommented by USSSETUP job: 2013/04/24 15:04:00
shell     stream tcp nowait IBMUSER  /usr/sbin/orshd orshd -LV
# Following line updated by USSSETUP job: 2013/04/24 15:04:00
login     stream tcp nowait IBMUSER  /usr/sbin/rlogind rlogind -m
# Following line added by USSSETUP job: 2013/04/24 15:04:00
ssh       stream tcp nowait IBMUSER  /usr/sbin/sshd sshd -i
#exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
# All users should use this configuration file
```

Figure 249. Example: Request body for a PUT request to the UNIX file /etc/inetd.conf

Example response

For a successful request, the HTTP response contains the following:
- Status code indicating that the request completed (status code 204)
- ETag that you can use on subsequent requests to test for changes to the UNIX file

```
204 No Content
Etag:  S8WNSD09SNSNE09B
Content-Type:  application/json; charset=UTF-8
Content-Language:  en-US
Date:  Fri, 07 Nov 2014 02:31:39 GMT
```

Create a UNIX file or directory

You can use this operation to create a UNIX file or directory.

HTTP method and URI path

```
POST /zosmf/restfiles/fs/<file-path>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/fs** indicates a UNIX file or directory.
- **<file-path>** is the name of the file or directory you are going to create.

Request Body

The request body to create a UNIX file or directory is shown in Request body to create a UNIX file or directory.

Table 222. Request body to create a UNIX file or directory

Field	Type	Description
type	String	The request type. This field supports the values: directory or dir to create a directory. The value: file is supported to create a file.
mode	String	Specifies the file or directory permission bits to be used in creating the file or directory. The characters used to describe permissions are: r: Permission to read the file w: Permission to write on the file x: Permission to execute the file -: No permission An example would be: rwxrwxrwx The nine characters are in three groups of three; they describe the permissions on the file or directory. The first group of 3 describes owner permissions; the second describes group permissions; the third describes other (or world) permissions.

Standard headers

None.

Custom headers

None.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `201` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For a successful create request, 201 Created is returned instead of returning an array of matching data sets.

Example request

In the following example, the POST method is used to create a UNIX file.

```
POST /zosmf/restfiles/fs/u/jiahj/text.txt HTTP/1.1

Request body:
{"type":"file","mode":"RWXRW-RW-"}
```

Example response

A sample response is shown in Create a UNIX file.

```
201 Created
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 30 Sep 2015 11:46:21 GMT
```

Figure 250. Example: Create a UNIX file

Example request

The POST method is used to create a UNIX directory.

```
POST /zosmf/restfiles/fs/u/jiahj/testDir HTTP/1.1
```

Request body

```
{"type":"directory","mode":"rwxr-xrwx"}
```

Example response

A sample response is shown in Create a UNIX directory.

```
201 Created
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Date:  Wed, 30 Sep 2015 10:50:21 GMT
```

Figure 251. Example: Create a UNIX directory

Delete a UNIX file or directory

You can use this operation to delete a UNIX file or directory.

HTTP method and URI path

```
DELETE /zosmf/restfiles/fs/<file-pathname>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/fs** indicates a UNIX file or directory.
- **<file-pathname>** is the name of the file or directory you are going to delete.

Request Body

None.

Standard headers

None.

Custom headers

X-IBM-Option: An optional parameter for deleting a directory. If it is not specified, only the empty directory can be deleted. If it is specified as recursive, it means all the files and sub-directories will be deleted.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `204` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see “Error handling” on page 384.

If the request is successfully executed, status code 204 indicates success and no content returned is returned.

Example request

In the following example, the DELETE method is used to delete a UNIX file.

```
DELETE /zosmf/restfiles/fs/u/jiahj/text.txt HTTP/1.1
```

Example response

A sample response is shown in Figure 252.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 16 Sep 2015 12:10:22 GMT
```

Figure 252. Example: Delete a UNIX file

Example request

The DELETE method is used to delete a UNIX directory.

```
DELETE /zosmf/restfiles/fs/u/jiahj/testDir HTTP/1.1
```

Example response

A sample response is shown in Delete a UNIX directory.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Wed, 16 Sep 2015 12:15:22 GMT
```

Figure 253. Example: Delete a UNIX directory

z/OS UNIX file utilities

You can use the z/OS UNIX file utilities to operate on a Unix System Services file or directory. Operations include: chmod, chown, chtag, copy, extattr, getfacl, move and setfacl.

HTTP method and URI path

```
PUT /zosmf/restfiles/fs/<file-path-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/fs** indicates a Unix System Services file system request
- ***<file-path-name>*** identifies the UNIX file/directory to be the target of the operation. This is required and must consist of a fully qualified path and file/directory name.

Headers

X-IBM-BPXK-AUTOCVT

This header is optional; use it to indicate how file auto conversion is handled when using the copy operation to copy text mode data sets to POSIX files, if you omit this header, the system default is taken.

'on' or 'all'

The target file is a candidate for automatic conversion if its TXTFLAG is tagged TEXT and the source data set is type TEXT.

'off'

The target file is not a candidate for automatic conversion

The header, Content-Type: application/json; charset={charset-name}, must be specified as well.

Request body

A JSON request document (content-type=application/json, character-encoding=UTF-8) must be supplied in one of the following forms:

Table 223. JSON request document

Function	Property	Description	Required
chmod	request	Indicates the function chmod.	Yes
	mode	The mode value, specified as the POSIX symbolic form or octal value (as a JSON string).	Yes
	links:"follow | suppress"	The default is 'follow' encountered links. This applies a mode change to the file or directory pointed to by any encountered links. 'suppress' is a mode change for the file or directory pointed to by any encountered symbolic links.	No
	recursive:true | false	The default is false. When 'true', the file mode bits of the directory and all files in the file hierarchy below it are changed. (chmod -R).	No

Table 223. JSON request document (continued)

Function	Property	Description	Required
chown	request	Indicates the function chown.	Yes
	owner	The user ID or UID (as a JSON string).	Yes
	group	The group ID or GID (as a JSON string).	No
	links:"follow | change"	The default is 'follow'. 'change' does not follow the link, but instead changes the link itself (chown -h).	No
	recursive:true | false	The default is false. When 'true', changes all the files and subdirectories in that directory to belong to the specified owner (and group, if :group is specified).chown -R)	No
chtag	request	Indicates function chtag.	Yes
	action:"set | remove | list"	The file tag action. If "set", the file will be tagged as specified in the "type" keyword. If "remove", any existing file tag will be removed. If "list", the existing tag information will be returned in a JSON response document. See "list action.	Yes
	type:"binary | mixed | text"	The default is "mixed" This option can only be specified if the action is "set".	No
	codeset	The default is 'follow'. 'change' does not follow the link, but instead changes the link itself (chown -h).	No
	links:"change | suppress"	The default is 'change' encountered links, applying a tag action to the file or directory pointed to by any encountered links. 'suppress' a tag action for the file or directory pointed to by any encountered	No
	recursive:true | false	The default is false. When 'true', tags all the files and subdirectories in that directory (chtag -R).	No
Note: If the 'list' action is specified, a response json document is returned listing the current tag information, For example: `{"stdout":["m ISO8859-1 T=off /tmp/file"]}` The -q and -v options are not supported.			

Table 223. JSON request document (continued)

Function	Property	Description	Required
copy	request	Indicates the function copy.	Yes
	from:*file-or-directory*	The file or directory to copy.	You can use either from:*file-or-directory* or from-dataset.
	from-dataset	**dsn** The fully qualified data set name. This is required.	
		member The data set member to copy. This is not required.	
		type One of "binary | executable | text". If not specified, the format of the data set will be checked to try to determine the type. This is not required.	
	overwrite:true|false	The default is true.	No
	recursive:true|false	The default is false. When 'true', copies all the files and subdirectories specified by source into a directory (cp -R). May not be specified with 'from-dataset'.	No
	links:"none|src|all"	The default is none. When 'src', follows symbolic links specified as source file/directory (cp -H). When 'all', follows symbolic links specified as source file/directory and those encountered in the tree traverse (cp -L). Cannot be specified with 'from-dataset'.	No
	preserve: "none|modtime|all"	The default is none, sets the modification time of the destination file to the present. When 'modtime', sets the modification and access time of each destination file to that of the corresponding sourcefile. (cp -m). When 'all', preserves the modification and access times as well as the file mode, file format, owner, and group owner (cp -p). May not be specified with 'from-dataset'.	No
Note: When from-dataset/type == text, and the header X-IBM-BPXK-AUTOCVT == ON | ALL,the cp "-O u" switch will be supplied to allow automatic conversion. If the from-dataset/type attribute is not specified, then no "-O u" switch will be applied and automatic conversion will not be available.			
extattr	request	Indicate the function extattr.	Yes
	set:”attrs”	One or more of the following: alps.	No
	reset:”attrs”	One or more of the following: alps.	No

Note: If neither set or reset are provided, a response json document is returned listing the attributes, For example:

```
{
  "stdout":
  ["/etc/inetd.conf",
  "APF authorized = NO",
  "Program controlled = NO",
  "Shared address space = YES",
  "Shared library=NO"]
}
```

The -F option is not supported.

Table 223. JSON request document (continued)

Function	Property	Description	Required
getfacl	request	Indicates the function getfacl.	Yes
	type:"access | dir | file"	The default is 'access', displays the access ACL entries for a file or directory (getfacl -a). 'dir' displays the directory default ACL entries (getfacl -d). If the target is not a directory, a warning is issued.	No
	user	The user ID or UID (as a JSON string), displays only the ACL entries for the specified types of access control lists (getfacl -a, -d, -f) which affects the specified user's access (getfacl -e user).	No
	use-commas:true | false	The default is 'false'. When true, displays each ACL entry,using commas to separate the ACL entries instead of newlines.	No
	suppress-header:true | false	The default is 'false'. When true, the comment header (the first three lines of each file's output) is not to be displayed (getfacl -m).	No
	suppress-baseacl:true | false	The default is 'false'. When true, displays only the extended ACL entries. Does not display the base ACL entries (getfacl -o).	No

Note: On completion of this request, a response json document is returned, For example:
`{"stdout":["#file:  /etc/inetd.conf","#owner: CFZSRV","#group: SYS1","user::rwx",`
`"group::rwx","other::rwx"]}`			

move	request	Indicates the function move.	Yes
	from	The file/directory to move.	Yes
	overwrite:true | false	The default is true.	No

Table 223. JSON request document (continued)

Function	Property	Description	Required
setfacl	request	Indicates the function setfacl.	Yes
	abort:true | false	The default is false. When true, aborts processing if an error or warning occurs. See the setfacl command documentation for complete documentation on the errors and warnings (setfacl -a).	No
	links:"follow | suppress"	The default is 'follow'. 'suppress' does not follow symbolic links. Because ACLs are not associated with symbolic links, nothing will happen if a symbolic link is encountered (setfacl -h). **Note:** At least one of the following four keywords must be specified. 'modify' and 'delete' may both be specified, but not with 'delete-type' and 'set'.	No
	delete-type	Delete all extended ACL entries by type (setfacl -D type): **`access`** Access ACL **`dir`** Directory default ACL **`file`** File default ACL **`every`** Every extended ACL for all ACL types that are applicable for the current path name. **Note:** The 'delete-type' keyword cannot be specified with 'set', 'modify' or 'delete'.	No
	set	sets (replaces) all ACLs with 'entries'. 'entries' represents a string of ACL entries. Refer to the setfacl command reference for the string format (setfacl -s entries). **Note:** The 'set' keyword cannot be specified with 'delete-type', 'modify' or 'delete'.	No
	modify	Modifies the ACL entries. 'entries' represents a string of ACL entries. Refer to the setfacl command reference for the string format. If an ACL entry does not exist for a user or group specified in 'entries', then it is created. If an ACL entry already exists for a user or group that was specified in 'entries', then it is replaced. **Note:** The 'modify' keyword cannot be specified with 'delete-type' or 'set'.	No
	delete	Deletes the extended ACL entries specified by 'entries'. 'entries' is a string of ACL entries. Refer to the setfacl command reference for the string format. If an ACL entry does not exist for the user or group specified, then you will not get an error. **Note:** The 'delete' keyword cannot be specified with 'delete-type' or 'set'.	No

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200 OK` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example

Refer to Figure 254 for an example of renaming a UNIX file.

```
Example request for renaming /etc/inetd.conf to /etc/inetd.conf.bak:

PUT https://zosmf1.yourco.com/zosmf/restfiles/fs/etc/inetd.conf.bak HTTP/1.1
Content-Type: application/json; charset=UTF-8

{"request":"move", "from":"/etc/inetd.conf"}
```

Figure 254. Example: Rename a UNIX file

List z/OS UNIX Filesystems

You can use the list z/OS UNIX filesystems operation to list all mounted filesystems, or the specific filesystem mounted at a given path, or the filesystem with a given Filesystem name.

HTTP method and URI path

```
GET /zosmf/restfiles/mfs/
GET /zosmf/restfiles/mfs/?path=file-path-name
GET /zosmf/restfiles/mfs/?fsname=file-system-name
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/mfs** indicates a Unix System Services filesystem(s) request.

  A trailing "/" may optionally be specified.

Query parameters

path
: This parameter identifies the UNIX directory that contains the files and directories to be listed. This parameter may not be specified if the 'fsname' parameter is specified. It can consist a directory or fully qualified path name in the UNIX file system structure. A fully qualified file name can be up to 1023 bytes long. You cannot use wildcard characters for this parameter.

fsname
: This parameter identifies the fully qualified filesystem name to be listed. For zFS filesystems, this is the data set name of the aggregate. This parameter may not be specified if the 'path' parameter is specified.

Headers

X-IBM-MAX-Items
: This header value specifies the maximum number of items to return. To request that all items be returned, set this header to 0. If you omit this header, or specify an incorrect value, up to 1000 items are returned by default.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200 OK` indicates success. Status code 404 indicates that the specified filesystem was not found. A JSON response document with no filesystem items will be returned. A status code of `4`*nn* or `5`*nn* indicates that an error has occurred. For more details, see “Error handling” on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See “Error report document” on page 454.

Example

Refer to Figure 255 for an example of renaming a data set and PDS member.
If more items than specified by X-IBM-Max-Items (or the default of 1000) match the request, then the following top-level attribute will be added to the top-level document:

```
"moreRows":true
```

```
JSON response document:
Request:
GET https://zosmf1.yourco.com/zosmf/restfiles/mfs/?path=/usr/local HTTP/1.1
GET https://zosmf1.yourco.com/zosmf/restfiles/mfs/?fsname=OMVS.USR.LOCAL.ZFS HTTP/1.1

JSON response document:

{"items":[
{ "name":"OMVS.USR.LOCAL.ZFS",
"mountpoint":"/usr/local",
"fstname":"ZFS",
"status":"active",
"mode":["acl","synchonly"],
"dev":52,
"fstype":1,
"bsize":1024,
"bavail":5615,
"blocks":9600,
"sysname":"SYSNAME1",
"readibc":2,
"writeibc":0,
"diribc":0
}
],"JSONversion":1}

For a non-specific request to list all filesystems, the following top-level attributes are also returned:

returnedRows - the number of filesystem items returned
totalRows - the total number of filesystems
```

Figure 255. List UNIX Filesystems

JSON response attributes:

These attributes are the mnt3_* values returned from the w_getmntent() C-Library API.

Create z/OS UNIX zFS Filesystem

You can use this operation to create a z/OS UNIX zFS Filesystem.

HTTP method and URI path

```
POST /zosmf/restfiles/mfs/zfs/<file-system-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/mfs/zfs** a Unix System Services filesystem request for a zFS aggregate. <*file-system-name*> is the filesystem (for zFS, the aggregate name) of the file system to be created. This is also the VSAM linear data set name.

Optional Query Parameters

timeout={secs}
The number of seconds to wait for the "zfsadm format" command to complete before timing out with Category/RC/REAS = 1/8/9 "Command timed out". The default if not specified is 20 seconds. If a greater value is used, the "X-IBM-Async-Threshold" header should be used to allow the application process long running transactions.

Input Content

Input content is a json document:

Table 224. Input Content

Property	Description
owner	Defaults to 755. This property is not required.
group	Defaults to 755. This property is not required.
perms	Defaults to 755. This property is not required.
cylsPri	Defaults to 0. This property is required.
cylsSec	Defaults to 0. This property is not required.
storageClass	This property is not required.
managementClass	This property is not required.
dataClass	This property is not required.
volumes	This property is not required.
JSONversion:1	This property is not required.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `201 OK` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example

Refer to Figure 256 for an example of creating UNIX Filesystems.

```
pfs->mfs
request:
POST https://zosmf1.yourco.com/zosmf/restfiles/pfs/zfs/HLQ.MYNEW.ZFS  HTTP/1.1
 Content-Type: application/json
 Content-Length: 86
 {
  "cylsPri":100,
  "cylsSec": 10,
  "volumes": [ "ZFS001", "ZFS002"],
  "JSONversion":1
 }

response:
201 Created
```

Figure 256. Create UNIX Filesystems

Delete z/OS UNIX zFS Filesystem

You can use the delete z/OS UNIX zFS Filesystem operation to delete an existing zFS filesystem. Access Method Services are used to delete the filesystem linear data set. The file system must not be allocated (attached or mounted) for this operation to succeed.

HTTP method and URI path

```
DELETE /zosmf/restfiles/mfs/zfs/<file-system-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/mfs/zfs** a Unix System Services filesystem request for a zFS aggregate. *<file-system-name>* is the filesystem (for zFS, the aggregate name) of the file system to be created. This is also the VSAM linear data set name.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200 OK` indicates success. Status code `204` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

For errors, the HTTP response includes error information as a JSON error report document. See "Error report document" on page 454.

Example

Refer to Figure 257 for an example of deleting UNIX Filesystems.

```
pfs->mfs
request:
DELETE https://zosmf1.yourco.com/zosmf/restfiles/pfs/zfs/HLQ.MYNEW.ZFS  HTTP/1.1

response:
204 No Content
```

Figure 257. Delete UNIX Filesystems

Mount a UNIX file system

You can use this operation to mount a UNIX file system on a specified directory.

HTTP method and URI path

```
PUT /zosmf/restfiles/mfs/<file-system-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/mfs**is used for managing file systems.
- **<file-system-name>** is the file system that you want to mount.

Request Body

The request body to mount a UNIX file system is shown in Request body to mount a UNIX file system.

Table 225. Request body to mount a UNIX file system

Field	Type	Description
action	String	Mount: you are going to mount an UNIX file system on the specified directory.
mount-point	String	Specifies the mount point you will be using to mount. It is usually a directory.
fs-type	String	Specifies the file system type that you are going to mount. The name must match the TYPE operand on a FILESYSTYPE statement in the BPXPRMxx parmlib member for the file system.
mode	String	Specifies the mode you intend to mount to the file system: Support Value: rdonly: read only rdwr: read write --all the values are case insensitive Default value: rdonly

Standard headers

None.

Custom headers

None.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See "Required authorizations" on page 384.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see "Error handling" on page 384.

If the request is successfully executed, status code 204 indicates success and no content is returned.

Example request

In the following example, the PUT method is used to mount a UNIX file system.

```
PUT /zosmf/restfiles/mfs/JIAHJ.ZOSMF.DRIVER.HFS HTTP/1.1

Request body
{"action":"mount","mount-point":"/u/jiahj","fs-type":"HFS","mode":"rdonly"}
```

Example response

A sample response is shown in Figure 258.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Thu, 17 Sep 2015 08:05:41 GMT
```

Figure 258. Example: Mount a UNIX file system

Unmount a UNIX file system

You can use this operation to unmount a UNIX file system on a specified directory.

HTTP method and URI path

```
PUT /zosmf/restfiles/mfs/<file-system-name>
```

where:

- **/zosmf/restfiles** specifies the z/OS data set and file REST interface
- **/mfs**is used for managing file systems.
- **<file-system-name>** is the file system you want to unmount.

Request Body

The request body to unmount a UNIX file system is shown in Request body to unmount a UNIX file system .

Table 226. Request body to unmount a UNIX file system

Field	Type	Description
action	String	The support value unmount is used to unmount a specified file system.

Standard headers

None.

Custom headers

None.

Query parameters

None.

Content type

The content type is application/json.

Required authorizations

See “Required authorizations” on page 384.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `204` indicates success. A status code of `4nn` or `5nn` indicates that an error has occurred. For more details, see “Error handling” on page 384.

If the request is successfully executed, status code 204 indicates success and no content returned is returned.

Example request

In the following example, the PUT method is used to unmount a UNIX file system.

```
PUT /zosmf/restfiles/mfs/JIAHJ.ZOSMF.DRIVER.HFS HTTP/1.1

Request body
{"action":"unmount"}
```

Example response

A sample response is shown in Unmount a UNIX file system.

```
204 No Content
Content-Type:  application/json; charset=UTF-8
Content-Length:  0
Date:  Thu, 17 Sep 2015 08:09:21 GMT
```

Figure 259. Example: Unmount a UNIX file system

JSON document specifications for z/OS data set and file REST interface requests

This section describes the contents of the JSON documents that are used with z/OS data set and file REST interface requests.

For more information about the properties described in these JSON documents, see *z/OS ISPF Services Guide*, SC34-4819.

The following JSON documents are described:

- "Data set list document"
- Data set list with attributes document
- PDS/PDSE member list with attributes document
- Create a sequential and partitioned data set document
- "File list document" on page 451
- Unix file and directory list with attributes document
- "Mount and unmount a file system document" on page 453
- Create a UNIX file document
- "Error report document" on page 454.

Data set list document

Table 227 shows the contents of the JSON data set list document.

For more information about these properties, see the dialog variables for the `YES|NO|PRT` parameter of **`LMDLIST`** (list data sets) in *z/OS ISPF Services Guide*, SC34-4819.

Table 227. Contents of the JSON data set list document

Property	Description	Required
items	An array where each element contains the following key:value pairs	Yes
dsname	Data set name.	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	No
JSONversion	Version number of this JSON document.	Yes
vol	Volume serial	No
used	Percentage of used tracks or pages (PDSE)	No
extx	Number of extents used, long format (5 bytes)	No
cdate	Creation date	No
edate	Expiration date	No
rdate	Date last referenced	No
migr	Whether the data set is migrated (YES or NO) based on the value of the `VOLUME_OF_MIGRATED_DATA_SETS` keyword in the ISPF configuration table. If the volume name of the data set matches the value of `VOLUME_OF_MIGRATED_DATA_SETS`, `ZDLMIGR` is set to YES, otherwise it is set to NO.	No
dsntp	Dsname type (PDS, LIBRARY, or ' ')	No
spacu	Space units	No

Table 227. Contents of the JSON data set list document (continued)

Property	Description	Required
mvol	Whether the data set is multivolume (Y) or not (N)	No
ovf	Space overflow indicator (YES or NO)	No
dsorg	Data set organization	No
recfm	Record format	No
lrecl	Logical record length	No
blksz	Block size	No
sizex	Data set size in tracks, long format (12 bytes)	No
catnm	Name of the catalog where the data set is located	No
dev	Device type	N0

Data set list with attributes document

"Data set list with attributes document" shows the attributes of the JSON data set list document.

For more information about these properties, see the dialog variables for the `YES|NO|PRT` parameter of **LMDLIST** (list data sets) in *z/OS ISPF Services Guide*, SC34-4819.

Table 228. X-IBM-Attributes=vol

Property	Description	Required
items	An array where each element contains the following key:value pairs **dsname** Data set name. **vol** Volume in which the data set resides.	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	No
JSONversion	Version number of this JSON document.	Yes

Table 229. X-IBM-Attributes=dsname

Property	Description	Required
items	An array where each element contains the following key:value pairs **dsname** Data set name.	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	Optional property; set to true when more rows can be returned.	No
totalRows	Total number of rows that match the request.	No
JSONversion	Version number of this JSON document.	Yes

Table 230. X-IBM-Attributes=base

Property	Description	Required
items	An array where each element contains the following key:value pairs Table 231	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	No
JSONversion	Version number of this JSON document.	Yes

Table 231. Items key:value pairs

Property	Description	Required
dsname	Data set name.	Yes
blksz	Block size.	No
catnm	Name of the catalog in which the data set was located.	No
cdate	Creation date.	No
dev	Device type.	No
dsntp	Percentage of used tracks or pages(PDSE).	No
dsorg	Data set organization.	No
edate	Expiration date.	No
extx	Number of extents used, long format (5 bytes).	No
lrecl	Logical record length.	No
migr	Whether the data set is migrated (YES or NO) based on the value of the VOLUME_OF_MIGRATED_DATA_SETS keyword in the ISPF configuration table. If the volume name of the data set matches the value of VOLUME_OF_MIGRATED_DATA_SETS, ZDLMIGR is set to YES, otherwise it is set to NO.	No
mvol	Whether the data set is multivolume (Y) or not (N).	No
ovf	Space overflow indicator (YES or NO).	No
rdate	Date last referenced.	No
recfm	Record format.	No
sizex	Data set size in tracks, long format (12 bytes).	No
spacu	Space units.	No
used	Percentage of used tracks or pages (PDSE).	No
vol	Volume serial.	No

PDS/PDSE member list with attributes document

Contents of the JSON PDS/PDSE member list document shows the contents of the JSON member list document.

For more information about these properties, see the dialog variables for the `YES|NO` parameter description of **`LMMFIND`** (find a library member) in *z/OS ISPF Services Guide*, SC34-4819.

Table 232. Contents of the JSON PDS/PDSE member list document

Property	Description	Required
items	An array where each element contains the following key:value pairs **member** Member name	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	No
JSONversion	Version number of this JSON document.	Yes

Table 233. X-IBM-Attributes=base and data set RECFM=F or V

Property	Description	Required
items	An array where each element contains the following key:value pairs. See Table 234.	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	No
JSONversion	Version number of this JSON document.	Yes

Table 234. Items key:value pairs for data set RECFM=F or V

Property	Description	Required
member	member name.	Yes
vers	Version number; a number from 1 to 99.	No
mod	Modification level; a number from 0 to 99.	No
c4date	Creation date in 4-character year format	No
m4date	Last change date in 4-character year format	No
cnorc	Current number of records; a number from 0 to 65 535.	No
inorc	Beginning number of records; a number from 0 to 65 535.	No
mnorc	Number of changed records; a number from 0 to 65 535.	No
mtime	Last change time; a character value in the format hh:mm.	No
msec	Seconds value of the last change time. This is a two character field.	No
user	User ID of last user to change the given member; an alphanumeric field with a maximum length of 7 characters.	No

Table 234. Items key:value pairs for data set RECFM=F or V (continued)

sclm	Indicates whether the member was last modified by SCLM or ISPF. A value of Y indicates the last update was made through SCLM. A value of N indicates that the last update was made.	No

Table 235. X-IBM-Attributes=base and data set RECFM=U

Property	**Description**	**Required**
items	An array where each element contains the following key:value pairs. See Table 236.	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	No
JSONversion	Version number of this JSON document.	Yes

Table 236. Items key:value pairs for data set RECFM=U

Property	Description	Required
member	member name.	Yes
ac	A 2-character field containing the authorization code of the member.	No
alias-of	An 8-character field containing the name of the real member that this member is an alias of. If the member is not an alias this field is blank.	No
amode	A 3-character field containing the AMODE of the member.	No
attr	A 20-character field containing the load module attributes. The attributes are 2-character strings separated by blanks. These strings can appear in the attribute string: NX Not executable OL Only Loadable OV Overlay RF Refreshable RN Reentrant RU Reusable SC Scatter Load TS Test	No
rmode	A 3-character field containing the RMODE of the member.	No
size	An 8-character field containing the load module size in hex.	No

Table 236. Items key:value pairs for data set RECFM=U (continued)

ttr	A 6-character field containing the TTR of the member.	No
ssi	An 8-character field containing the SSI information for a load module.	No

Create a sequential and partitioned data set document

Table 237shows the contents of the JSON sequential and partitioned data set document.

For more information about these properties, see the dialog variables for the YES|NO parameter description of **LMMFIND** (find a library member) in *z/OS ISPF Services Guide*, SC34-4819.

Table 237. Contents of the create a sequential and partitioned data set document

Property	Description	Required
volser	Volume serial number of the device a data set will reside on.	No
unit	Unit name of the device that the data set will reside on.	No
dsorg	Unit name of the device that the data set will reside on.	No
alcunit	Unit of space allocation.	No
primary	Primary space allocation for the data set.	No
secondary	Secondary space allocation for the data set.	No
dirblk	Number of directory blocks for a partitioned data set.	No
avgblk	Specifies the unit of space allocations to be blocks and sets the average block length.	No
recfm	Record format of a data set: Current support: F: fixed V: variable U: undefined B: block FB: fixed blocked VB: variable blocked	No
blksize	Block size of the data set.	No
lrecl	Record length of data set.	No
storclass	Specifies the storage class of system managed storage.	No
mgntclass	Specifies the management class of the data set.	No
dataclass	Specifies the data class of the data set.	No

File list document

Table 238 on page 452 shows the contents of the JSON file list document for UNIX files.

For more information about these properties, see the column-list parameter of the **DIRLIST** (directory list service) in *z/OS ISPF Services Guide*, SC34-4819.

Table 238. Contents of the JSON file list document

Property	Description
items	JSON array of member entries containing the properties that follow in the remainder of this table.
filename	File name
returnedRows	Number of rows that were returned for this request.
moreRows	Optional property; set to true when more rows can be returned.
totalRows	Total number of rows that match the request.

Unix file and directory list with attributes document

Contents of the unix file and directory shows the contents of the JSON file list document for UNIX files.

For more information about these properties, see the `column-list` parameter of the **`DIRLIST`** (directory list service) in *z/OS ISPF Services Guide*, SC34-4819.

Table 239. Contents of the unix file and directory

Property	Description	Required
items	An array where each element contains the following key:value pairs. See Unix items key:value pairs.	Yes
returnedRows	Number of rows that were returned for this request.	Yes
moreRows	True, if more rows are available to return.	No
totalRows	Total number of data sets found matching the dslevel and volser criteria. If you specify ",total" as suffix in X-IBM-Attributes header, like "dsname,total", or "base,total", or "vol,total".	Yes
JSONversion	Version number of this JSON document	Yes

Table 240. Unix items key:value pairs

Property	Description	Required
name	File or directory name.	Yes
mode	indicating the permissions.	No
size	For regular files, the file's size in bytes. For other kinds of files, the value of this field is unspecified.	No
uid	The numeric user ID (UID) of the file's owner.	No
user	The user name of the file's owner got by UID.	No
gid	The numeric group ID (GID) of the file's group.	No
group	The group name of the file's group got by GID.	No
mtime	The most recent time the contents of the file were changed.	No
target	If the file is synlink, this indicates the really file/directory	No

Mount and unmount a file system document

Contents of the Mount and unmount a file system document shows the contents of the JSON file list document for UNIX files.

For more information about these properties, see the `column-list` parameter of the **`DIRLIST`** (directory list service) in *z/OS ISPF Services Guide*, SC34-4819.

Table 241. Contents of the Mount and unmount a file system document

Property	Description	Required
action	Current support value: mount: you are going to mount an UNIX file system on the specified directory. unmount: you are going to unmount a specified directory.	Yes
mount-point	Specify the mount point you are going to mount/unmount, generally, it is directory.	No
fs-type	Specify the file system type you are going to mount, --the name must match the TYPE aperand on a FILESYSTYPE statement in the BPXPRMxx parmlib member for the file system. For the mount action, this is required; but it is not required for the unmount action.	No
mode	Specify the mode you intend to mount the file system: Support Value: rdonly: read only rdwr: read write --all the values are case insensitive	No

Create a UNIX file document

Contents of the create a UNIX file document shows the contents of the JSON file list document for UNIX files.

For more information about these properties, see the `column-list` parameter of the **`DIRLIST`** (directory list service) in *z/OS ISPF Services Guide*, SC34-4819.

Table 242. Contents of the create a UNIX file document

Property	Description	Required
type	The request type, support value: directory or dir: you are going to create a directory. file: you are going to create a file.	No

Table 242. Contents of the create a UNIX file document (continued)

Property	Description	Required
mode	The mode specifies the file/directory permission bits to be used in creating the file/directory. **rwxrwxrwx** The 9 characters are in three groups of 3; they describe the permissions on the file. The first group of 3 describes owner permissions; the second describes group permissions; the third describes other (or "world") permissions. Characters that might appear are: r: Permission to read the file w: Permission to write on the file x: Permission to execute the file -: No such a permission	No

Error report document

Table 243 shows the contents of the JSON error report document for z/OS data set and file REST interface requests.

Table 243. Contents of the JSON error report document

Property	Description
category	Error category. This field is integer data type.
rc	Return code. This field is integer data type.
reason	Reason code. This field is integer data type.
message	Message that describes the error.
details	(optional) Array of strings containing additional message details.
stack	Stack trace of the exception.

For the meanings of the category, rc, and reason fields, see "Error reporting categories" on page 455.

Error reporting categories

This section describes the error categories and associated error codes that can be returned in the JSON error report document for z/OS data set and file REST interface requests. This document is described in "Error report document" on page 454.

Categories

Table 244 shows the error categories that are defined for errors returned in z/OS data set and file REST interface operations.

Table 244. Error categories for z/OS data set and file REST interface operations

Category	Ordinal Value	Description	Where the error details are described
Service	1	Errors produced or detected in the service layer.	"Category 1 — Service error"
Message queue	2	Errors produced or detected by the message queue.	"Category 2 — Message queue error" on page 457
CEA	3	Errors produced or detected by the common event adapter (CEA) interface.	"Category 3 — Common event adapter (CEA) error" on page 458
ISPF	4	Errors produced or detected by the Interactive System Productivity Facility (ISPF) interface.	"Category 4 — ISPF error" on page 458
CSI	5	Errors produced or detected by the catalog search interface (CSI).	"Category 5 — Catalog Search Interface (CSI) error" on page 458
Read or write function	6	Errors returned from an attempted read or write request.	"Category 6 — Read or write error" on page 459
JSON	7	Errors produced or detected when parsing JSON.	"Category 7 - JSON Parser Conditions" on page 460
XL C/C++	8	Errors produced or detected by z/OS XL C/C++	"Category 8 - z/OS XL C/C++ Conditions" on page 460
Unexpected	16	Unexpected errors detected.	"Category 9 — Unexpected error" on page 460

Category 1 — Service error

Table 245 shows the possible conditions for this error category.

Table 245. Category 1 errors for z/OS data set and file REST interface operations

rc	reason	message	Description
4	1	`%s`	The request specified a URL that is not valid. %s indicates which parts of the URL are invalid, such as path information, data set name and volume serial.
4	2	`Invalid data set name length.`	The request specified a data set name length that is not valid. Note that data set names cannot exceed 44 characters in length.
4	3	`The specific data set name is invalid.`	The request specified a data set name (dslevel) that is not valid.

Table 245. Category 1 errors for z/OS data set and file REST interface operations (continued)

rc	reason	message	Description
4	4	Retrieving USS files should start with absolute path.	The requested file cannot be retrieved because the request is missing an absolute path name, which is required. When retrieving UNIX files, the file path must be an absolute (or fully-qualified) path name, rather than a relative (or partially-qualified) path name.
4	5	Length of USS path is invalid.	The request specified a UNIX file path length that is not valid.
4	6	The specific path is invalid.	The request specified a UNIX file path that is not valid.
4	7	The specific volser is invalid.	The request specified a volume serial (volser) that is not valid.
4	8	File not found.	The request specified a file that does not exist.
4	9	Incorrect content type.	The request specified an unsupported content type.
4	10	Unrecognized HTTP method.	The request is not recognized as a supported HTTP method.
4	11	POST method is not supported.	The POST method was specified, however POST is not a supported HTTP method.
4	12	Incorrect attribute was specified.	The request contained one or more attributes that are not valid.
4	13	Incorrect parameter was specified.	The request contained one or more parameters that are not valid.
4	14	The specific JSON data in the request is invalid.	JSON data in the request is invalid.
4	15	The size of the data to be written is invalid.	The size of the data to be written is invalid.
4	16	Request content-length must be specified. Request content-length was too long. Request content was longer than specified content-length.	Only used in ValidateJsonServlet.C.
4	17	Request document contained invalid JSON.	Not currently used.
4	18	Member name is not valid.	Member name is not valid.
4	19	Unix file/directory exists.	Unix file/directory already exists.
8	1	Unable to get JSON Response.	Not currently used.
8	2	Unable to get JSON Response Table.	Not currently used.
8	3	Unable to load ISPEXEC.	The ISPF service route ISPEXEC could not be loaded.
8	4	Unable to load TSO servlet mappings.	The servlet dispatcher was not able to load the servlet mappings JSON file.
8	5	A servlet-mapping was not found for servlet-path.	A servlet mapping matching the request could not be found. Indicates a servlet-mapping configuration error.
8	6	ExceptionServlet threw an exception after commit. Servlet failed but could not send error response.	An exception was thrown by the servlet.

Table 245. Category 1 errors for z/OS data set and file REST interface operations (continued)

rc	reason	message	Description
8	7	`TsoServlet already committed.`	An exception was thrown by the servlet after its output stream header was committed.

Category 2 — Message queue error

Table 246 shows the possible conditions for this error category.

Table 246. Category 2 errors for z/OS data set and file REST interface operations

rc	reason	message	Description
12	1	`The message queue cannot be created.`	The message queue cannot be created.
4	1	`Timeout`	A timeout occurred when receiving a message from the message queue. MSG_QUEUE_PROTOCOL_ERROR_TIMEOUT
4	2	`Received unexpected msgType=nn`	Received unexpected message. MSG_QUEUE_PROTOCOL_ERROR_UNEXPECTED
4	3	`ServletDispatcher failed.`	Back-end ServletDispatcher failed with the reason <reason>, for example, TSO Prompt was received when TsoServletResponse was expected. MSG_QUEUE_PROTOCOL_ERROR_ENDED
4	4	`Queue full while sending.`	Message queue is full while sending the specific <message type>. MSG_QUEUE_PROTOCOL_ERROR_FULL
4	5	`Illegal state.`	The message queue is in an illegal state. MSG_QUEUE_PROTOCOL_ERROR_ILLEGAL_STATE
5	1	`Error parsing TsoServletResponse.`	Error occurred when paring TsoServletResponse. MSG_QUEUE_JSON_PARSE_ERROR
5	2	`Error serializing TsoServletRequest.`	Error occurred when serializing TsoServletRequest. MSG_QUEUE_JSON_SERIALIZE_ERROR
12	1	`The Message queue cannot be created.`	The message queue cannot be created.
12	2		Not used currently.
12	3	`Message queue size is less than minimum.`	Message queue size is less then the required minimum size. MSG_QUEUE_ERROR_SIZE_ERROR
12	4	`Message prefix bytes are too short.`	Message prefix bytes are shorter than expected. MSG_QUEUE_ERROR_DECODING
16	n(errno)	`Varies.`	This is an error from msgsnd()/msgrcv()/etc. The errno will be used as the reason. MSG_QUEUE_SYS_ERROR

Category 3 — Common event adapter (CEA) error

Table 247 shows the possible conditions for this error category.

Table 247. Category 3 errors for z/OS data set and file REST interface operations

rc	reason	message	Description
12	1	`TSO launcher exception: Client is not authorized for instrumentation.`	The requestor lacks sufficient authority to access the requested common event adapter (CEA) service.
12	2	`TSO launcher exception: Error occurred.`	The requested CEA service encountered an error.
12	3	`TSO launcher exception: CEA address space is not available.`	The CEA address space is not active or is not available.
12	4	`TSO launcher exception: TSO address space cannot be created.`	The TSO/E address space cannot be created because a required system resource is not available.
12	5	`CeaTsoEnd request failed.`	An error occurred ending a TSO Address Space. CEA TSO Reason code = 4103. CEA_ERROR_END_TSO_FAILED
12	6		Not currently used. CEA_ERROR_CMD_NOT_FOUND

Category 4 — ISPF error

Table 248 shows the possible conditions for this error category.

Table 248. Category 4 errors for z/OS data set and file REST interface operations

rc	reason	message	Description
n	m	varies	The return and reason code values match the return and reason code values that are set by the ISPF service.

Category 5 — Catalog Search Interface (CSI) error

Table 249 shows the possible conditions for this error category.

Table 249. Category 5 errors for z/OS data set and file REST interface operations

rc	reason	message	Description
n	m	varies	The return and reason code values match the return and reason code values that are set by the consolidated software inventory (CSI) service.

Category 6 — Read or write error

Table 250 shows the return and reason codes that can be set for a read or write request.

Table 250. Category 6 errors for z/OS data set and file REST interface operations

rc	reason (hex)	message	Description
8	201	<methodName> failed	Unable to open a data set or member, <methodName> like fopen() or freopen() failed. RW_ERROR_OPEN_FAILED
8	202	<methodName> failed	Unable to close a data set or member,<methodName> like fclose() failed. If occurs for a write/put operation, the data set contents are not predictable. failedRW_ERROR_CLOSE_FAILED
8	204	Client ETag does not match the current ETag for the data set.	The attempt to write to the data set failed because the supplied ETag does not match the current ETag of the requested data set. This mismatch indicates that the data set content was modified in the time since the caller obtained the ETag. RW_ERROR_DS_ETAG_NOT_MATCHED
8	208	.<methodName> error	An error, <methodName> like fread(), occurred during I/O to a data set or Unix file. RW_ERROR_IO
8	20C	Member not found.	The member cannot be located in the partitioned data set. Perhaps the data set name or member name was incorrectly specified. RW_ERROR_MBR_NOT_FOUND
8	0000020A	Dynamic allocation failed.	This RC is combined with S99ERROR in the high halfword. RW_ERROR_DS_DYNALLOC_ERR
8	1708020A	ISPF LMINIT - data set not found.	The specified data set cannot be found. Perhaps the data set name or member name was incorrectly specified. RW_ERROR_DS_NOT_FOUND
8	varies	Varies.	For UNIX file I/O errors, the reason code consists of the errno in the high order 16 bits and the errno2 in the low order 16 bits.
8	5E30062	File not found.	The specified UNIX file cannot be found. Perhaps the data set name or member name was incorrectly specified. RW_ERROR_FS_NOT_FOUND
8	5B6F0002	Client is not authorized for file access.	The request for the UNIX file failed because the caller does not have sufficient authority to access the file. RW_ERROR_FS_AUTH

Table 250. Category 6 errors for z/OS data set and file REST interface operations (continued)

rc	reason (hex)	message	Description
8	406	`Client ETag does not match the current ETag for the file.`	The attempt to write to the UNIX file failed because the supplied ETag does not match the current ETag of the requested file. This mismatch indicates that the file content was modified in the time since the caller obtained the ETag. RW_ERROR_FS_ETAG_NOT_MATCHED

Category 7 - JSON Parser Conditions

Category 7 JSON parser conditions shows the possible conditions for this error category.

Table 251. Category 7 JSON parser conditions

rc	reason	message	Description
n	m	`Varies.`	The rc and reason are set from the low level JSON Parser return code and reason code.

Category 8 - z/OS XL C/C++ Conditions

Category 8 z/OS XL C/C++ Conditions shows the possible conditions for this error category.

Table 252. Category 8 z/OS XL C/C++ Conditions

rc	reason	message	Description
n	m	`Varies.`	The rc and reason are set from the low level z/OS XL C/C++ return code and reason code.

Category 9 — Unexpected error

Table 253 shows the possible conditions for this error category.

Table 253. Category 9 errors for z/OS data set and file REST interface operations

rc	reason	message	Description
16	1	`Server error occurred.`	For details about the exception, see the stack property of the JSON error report document, which is described in "Error report document" on page 454.

z/OS jobs REST interface

The z/OS jobs REST interface is an application programming interface (API) implemented through industry standard Representational State Transfer (REST) services. A set of REST services is provided for working with batch jobs on a z/OS system, as described in this topic.

Table 254 lists the operations that the z/OS jobs REST interface services provide.

Table 254. Operations provided through the z/OS jobs REST interface services.

Operation	HTTP method and URI path
"Obtain the status of a job" on page 466	GET /zosmf/restjobs/jobs/<*jobname*>/<*jobid*> GET /zosmf/restjobs/jobs/<*correlator*>
"List the jobs for an owner, prefix, or job ID" on page 468	GET /zosmf/restjobs/jobs[?<*parms*>]
"List the spool files for a job" on page 471	GET /zosmf/restjobs/jobs/<*jobname*>/<*jobid*>/files GET /zosmf/restjobs/jobs/<*correlator*>/files
"Retrieve the contents of a job spool file" on page 473	GET /zosmf/restjobs/jobs/<*jobname*>/<*jobid*>/files/<*nnn*>/records GET /zosmf/restjobs/jobs/<*correlator*>/files/<*nnn*>/records GET /zosmf/restjobs/jobs/<*jobname*>/<*jobid*>/files/JCL/records GET /zosmf/restjobs/jobs/<*correlator*>/files/JCL/records
"Submit a job" on page 476	PUT /zosmf/restjobs/jobs[/-<*JESB*>]
"Hold a job" on page 481	PUT /zosmf/restjobs/jobs/<*jobname*>/<*jobid*> PUT /zosmf/restjobs/jobs/<*correlator*>
"Release a job" on page 484	PUT /zosmf/restjobs/jobs/<*jobname*>/<*jobid*> PUT /zosmf/restjobs/jobs/<*correlator*>
"Change the job class" on page 487	PUT /zosmf/restjobs/jobs/<*jobname*>/<*jobid*> PUT /zosmf/restjobs/jobs/<*correlator*>
"Cancel a job" on page 490	PUT /zosmf/restjobs/jobs/<*jobname*>/<*jobid*> PUT /zosmf/restjobs/jobs/<*correlator*>
"Cancel a job and purge its output" on page 493	DELETE /zosmf/restjobs/jobs/<*jobname*>/<*jobid*> DELETE /zosmf/restjobs/jobs/<*correlator*>

Processing overview

The z/OS jobs REST interface services can be invoked by any HTTP client application, running on the z/OS local system or a remote system.

Your program (the client) initiates an HTTP request to the z/OS jobs REST interface. If the interface determines that the request is valid, it performs the requested service. After performing the service, the z/OS jobs REST interface creates an HTTP response. If the request is successful, this response takes the form of an HTTP 2*nn* response and, if applicable, a result set that is passed back to your program.

Depending on which service was requested, the result set might be returned in a format that requires parsing by your program, for example, a JSON object. In other cases, results might be returned in another format, such as plain text or binary data. If the request is not successful, the response consists of a non-OK HTTP response code with details of the error provided in the form of a JSON object. The contents of the JSON objects are described in "JSON document specifications for z/OS jobs REST interface requests" on page 496.

Resource URLs

The URLs of the z/OS jobs REST interface have the format that is shown in Figure 260:

```
https://{host}:{port}/zosmf/restjobs/jobs/{-jesb/}{resource}?{parm}
```

Figure 260. Format of resource URLs for z/OS jobs REST interface.

where:

- `"https://{host}:{port}"` specifies the target system address and port
- `"/zosmf/restjobs/jobs/"` identifies the z/OS jobs REST interface.
- `"JESB"` optionally specifies a secondary JES subsystem, if one is to be used to process the request. If you omit this value, the request is processed by the primary JES subsystem.
- `"{resource}?{parm}"` represents the resource, such as a job name and job ID, and optionally one or more parameters, to qualify the request.

HTTP methods

The z/OS jobs REST interface provides the following HTTP methods:

GET
Retrieves information about jobs that are running on the z/OS system.

PUT
Updates job information on the z/OS system, or sets attributes and performs actions on jobs.

DELETE
Removes jobs from the z/OS system.

Some situations might require the use of the POST method; see "Usage considerations for the z/OSMF REST services" on page 2.

Supported HTTP versions

z/OS jobs REST interface supports requests in either of the following protocols: HTTP/1.0 or HTTP/1.1

Content types

The data that is sent or returned by the HTTP methods has one of the following content types:

- Application/octet-stream (`Content-Type: application/octet-stream`) is used for data that is sent or returned in an uninterpreted format, such as a job that is submitted, or binary data or records that are obtained from a z/OS job spool file.
- JSON (`Content-Type: application/json`) is used for sent data and returned data; for the detailed format of each JSON object, see the description for each operation.
- Plain text (`Content-Type: plain/text`).

Error handling

For errors that occur during the processing of a request, the API returns an appropriate HTTP status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. For example, `HTTP/1.1 400 Bad Request` or `HTTP/1.1 500 Internal Server Error`.

In addition, some errors might also include a returned JSON object that contains a message that describes the error. You can use this information to diagnose the problem or provide it to IBM Support, if required. For the contents of the error report document, see "Error report document" on page 500.

The following HTTP status codes are valid:

HTTP 200 OK
Success.
HTTP 201 Created
Request was successful, and, as a result, a resource was created.
HTTP 202 Accepted
Request was received and accepted for processing, but the processing has not yet completed.
HTTP 400 Bad request
Request contained incorrect parameters.
HTTP 500 Internal server error
Programming error.

Synchronous support for the job modify operations

The z/OS jobs REST interface includes services that you can use to perform job modify operations, as shown in Table 255.

Table 255. Job modify operations provided through the z/OS jobs REST interface services.

Operation	Where described
Hold a job.	"Hold a job" on page 481
Release a job.	"Release a job" on page 484
Change the job class.	"Change the job class" on page 487
Cancel a job.	"Cancel a job" on page 490
Delete a job (cancel a job and purge its output).	"Cancel a job and purge its output" on page 493

These services can be run synchronously, if coded to use the latest version of the service. To request synchronous processing, set the "version" property in your request to `2.0`. Here, the system will attempt to process the request synchronously, if such processing is supported on the target JES subsystem. Synchronous processing is supported for JES2 subsystems only. When the target subsystem is JES3, a synchronous request is ignored and the service is performed asynchronously.

Generally, the differences in processing are as follows:

- For an asynchronous request, z/OSMF returns control to the caller immediately. However, to verify that the initial request was performed, the caller must subsequently issue the service described in "Obtain the status of a job" on page 466. Asynchronous processing is the default for all z/OS jobs REST interface services.
- For a synchronous request, z/OSMF does not return control to the caller until the requested action is performed and results of the request are available to be returned to the caller. Here, the JSON job feedback document provides more information about the success or failure of the request; see "Job feedback document" on page 498.

If your program does not require feedback on the results of requested actions, you can use these services asynchronously.

Due to timing behavior, if you submit a job and immediately issue a synchronous request for the same job, you might receive the error message `"No job found for reference"` in the JSON error report document (category 6, return code 4, reason code 10). To avoid this occurrence, it is recommended that you allow a small amount of time to pass between a job submit request and a subsequent job modify request.

Required system setup

The z/OS jobs REST interface services require that the System REXX (SYSREXX) component be set up and active on your z/OS system. For information, see *IBM z/OS Management Facility Configuration Guide*.

Required authorizations

Generally, your user ID requires the same authorizations for using the z/OS jobs REST interface services as when you perform these operations through a TSO/E session on the system. For example, submitting a job through the z/OS jobs REST interface requires that your user ID be authorized to run jobs on the system and be able to access any protected resources that the job might require.

In addition, your user ID requires authorization to the z/OSMF SAF profile prefix on the target z/OS system, as follows:

- READ access to <*SAF_PREFIX*> in the APPL class
- READ access to the <*SAF_PREFIX*>.izuUsers profile in the EJBROLE class.

By default, the z/OSMF SAF profile prefix is IZUDFLT.

If you are using z/OS jobs REST interface services to perform job modify operations, your user ID must be authorized to the appropriate resources in the JESJOBS class, as shown in Table 256.

Table 256. JESJOBS class authorizations needed for performing job modify operations

Operation	JESJOBS resource	Access required
Hold a job	HOLD.nodename.userid.jobname	UPDATE
Release a job	RELEASE.nodename.userid.jobname	UPDATE
Change the job class	MODIFY.nodename.userid.jobname	UPDATE
Cancel a job	CANCEL.nodename.userid.jobname	ALTER
Delete a job (cancel a job and purge its output)	CANCEL.nodename.userid.jobname	ALTER

For more information about JESJOBS class, see *z/OS Security Server RACF Security Administrator's Guide*.

If run asynchronously, these services also require that your user ID be authorized to the Common Information Model (CIM) server and permitted to the JES2-JES3Jobs CIM provider. CIM includes jobs (CFZSEC and CFZRCUST) to help you configure the CIM server, including security authorizations and file system customization. For information, see the chapter on CIM server quick setup and verification in *z/OS Common Information Model User's Guide*, SC33-7998.

Where applicable, additional authorization requirements are noted in the descriptions of the individual z/OS jobs REST interface services. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Requesting asynchronous job notifications

You can use the asynchronous job notifications function of z/OSMF to allow your programs to be notified when submitted jobs complete. With this function, the program that submits the job through the z/OS jobs REST interface services PUT method specifies a URL when submitting the job. When the job ends, z/OSMF returns an HTTP message to the URL location, indicating the job completion status. The data returned is in the form of a JSON document.

The asynchronous job notifications function is available for JES2 systems only; it is not available for JES3 systems.

The key requirement is that you must create a subscription to the Common Information Model (CIM) jobs indication provider for your system. Also, if the job notifications will require a secure network connection, you must enable an SSL connection between the client application and the server, including the sharing of digital certificates. For instructions on enabling the asynchronous job notifications function, see *IBM z/OS Management Facility Configuration Guide*.

Enabling browser login through a client certificate

It is possible to run the z/OS jobs REST interface services directly from a web browser. Here, you must first authenticate to z/OSMF through your browser. In z/OSMF, authentication is typically done by entering your user ID and password at the Welcome page. However, it is also possible to log in with a client certificate, if your installation favors this approach. With a client certificate, you can access z/OSMF through your browser without having to supply a user ID and password.

In client certificate authentication, the certificate is stored in the browser itself. When you login to z/OSMF, the server requests the certificate from your browser. If your browser stores more than one certificate, you might be prompted to select the correct one to use with z/OSMF. Otherwise, your browser sends the certificate to z/OSMF. After z/OSMF identifies you as the owner of the key that is associated with the certificate, a secure connection is established.

If z/OSMF does not accept your client certificate, z/OSMF displays the Welcome page for you to enter your user ID and password.

If your installation plans to enable client certificate login for the z/OS jobs REST interface services, understand that it is your responsibility to create the certificate and manage its distribution to users. It is recommended that you ensure that users have browsers that support importing a certificate.

For information about creating digital certificates, see *z/OS Security Server RACF Security Administrator's Guide*.

Error logging

Errors from the z/OS jobs REST interface services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Enabling traces for IBM analysis

For diagnostic purposes, your installation might be asked by IBM Support to enable tracing for the z/OS jobs REST interface. For information, see Appendix A, "Enabling tracing for the z/OS jobs REST interface," on page 747

Obtain the status of a job

You can use this operation to obtain the status of a batch job on z/OS.

HTTP method and URI path

```
GET /zosmf/restjobs/jobs/<jobname>/<jobid>
GET /zosmf/restjobs/jobs/<correlator>
```

where:

Custom headers

None.

Query parameters

None.

Required authorizations

See "Required authorizations" on page 464.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, note that this request can be directed to a secondary JES subsystem. To do so, use the following URL format:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid
```

where *JESB* is the name of the secondary JES subsystem. A request to a secondary JES subsystem must include the job name and job id, rather than a job correlator.

Expected response

On completion, the z/OS jobs REST interface returns an HTTP response with a JSON job document. For the contents, see "Job document" on page 496.

For errors, z/OS jobs REST interface returns an appropriate HTTP status code and error information as a JSON error report document. See "Error report document" on page 500.

Example request

The following request obtains the status for the job BLSJPRMI, job ID STC00052:

```
GET /zosmf/restjobs/jobs/BLSJPRMI/STC00052 HTTP/1.1
Host: zosmf1.yourco.com
```

Example response

A sample response is shown in Figure 261.

```
HTTP/1.1 200 OK
Date: Thu, 09 Dec 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
  "jobid":"STC00052",
  "jobname":"BLSJPRMI",
  "subsystem":"JES2",
  "owner":"IBMUSER",
  "status":"OUTPUT",
  "type":"STC",
  "class":"STC",
  "retcode":"CC 0000",
  "url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/S0000052SY1.....CE35BDE8.......%3A",
  "files-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/S0000052SY1.....CE35BDE8.......%3A\/files",
  "job-correlator":"S0000052SY1.....CE35BDE8.......:",
  "phase":20,
  "phase-name":"Job is on the hard copy queue"
}
```

Figure 261. Example: Returned job status

List the jobs for an owner, prefix, or job ID

You can use this operation to list the jobs for an owner, prefix, or job ID.

HTTP method and URI path

```
GET /zosmf/restjobs/jobs[?<parms>]
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<parms>*** are optional parameters that you can use to qualify the request. For a list of the supported parameters, see "Query parameters."

Custom headers

None.

Query parameters

You can specify one or more of the following optional query parameters on this request:

owner
: User ID of the job owner whose jobs are being queried; the default is the z/OS user ID. Folded to uppercase; cannot exceed eight characters.

prefix
: Job name prefix; defaults is *. Folded to uppercase; cannot exceed eight characters.

jobid
: Job ID. Folded to uppercase; cannot exceed eight characters. This query parameter is mutually exclusive with user-correlator.

max-jobs
: Maximum number of jobs returned. The value must be between 1 and 1000, inclusive. If this parameter is not specified, or is specified incorrectly, the default value of 1000 is used.

user-correlator
: The user portion of the job correlator. This value is 1 – 32 characters in length, where the first character must be uppercase alphabetic (A-Z) or special ($, #, @). The remaining characters (up to 31) can be any combination of uppercase alphabetic, numeric (0-9), or special. Blank characters are not supported.

  This query parameter is mutually exclusive with jobid.

  This value is processed by the JES2 subsystem only; the JES3 subsystem does not process the correlator and, instead, indicates zero job matches. For a system with JES3 as the primary subsystem, and one or more JES2 secondary subsystems, the primary JES3 subsystem does not process the correlator, however, the JES2 secondary subsystems can process the correlator.

Observe the following conventions:

- Query parameters are optional; you can specify one or more query parameters, as needed.
- You use a question mark (?) to separate the first query parameter from the resource.
- To specify multiple query parameters in combination, use an ampersand (&).
- Wildcard characters are permitted in the owner and prefix query parameter values. Use an asterisk (*) for multiple characters, and a question mark (?) for a single character.

Required authorizations

See "Required authorizations" on page 464.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, observe the following considerations for this request:

- The ordering of the jobs returned is not predictable.
- If the maximum number of jobs is returned, no indication is provided for whether more jobs remain to be retrieved.
- This request can be directed to a secondary JES subsystem. To do so, use the following request format:

```
https://host:port/zosmf/restjobs/jobs/-JESB
https://host:port/zosmf/restjobs/jobs/-JESB?owner=owner
```

  where *JESB* is the name of the secondary JES subsystem in these examples.
- To list the jobs for a job ID on a secondary JES subsystem, you must specify the job id, rather than a job correlator. For example:

```
https://host:port/zosmf/restjobs/jobs/-JESB?jobid=jobid
```

Expected response

On completion, the z/OS jobs REST interface returns an HTTP response with an array of matching jobs, each as a JSON job document. For the contents, see "Job document" on page 496.

For errors, z/OS jobs REST interface returns an appropriate HTTP status code and error information as a JSON error report document. See "Error report document" on page 500.

Example request

In the following example, the GET method is used to list the jobs that are owned by IBMUSER and have a job name prefix beginning with TESTJOB:

```
GET /zosmf/restjobs/jobs?owner=IBMUSER&prefix=TESTJOB* HTTP/1.1
Host: zosmf1.yourco.com
```

Example response

A sample response is shown in Figure 262 on page 470.

```
HTTP/1.1 200 OK
Date: Fri, 17 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

[
{"jobid":"JOB00023","jobname":"TESTJOB2","subsystem":null,"owner":"IBMUSER",
"status":"OUTPUT","type":"JOB","class":"A","retcode":"CC 0000",
"url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB2\/JOB00023",
"files-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB2\/JOB00023\/files"},
{"jobid":"JOB00024","jobname":"TESTJOB3","subsystem":null,"owner":"IBMUSER",
"status":"OUTPUT","type":"JOB","class":"A","retcode":"ABEND S000",
"url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB3\/JOB00024",
"files-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB3\/JOB00024\/files"}
]
```

Figure 262. Example: Returned list of the jobs for a specific owner and job name prefix

List the spool files for a job

You can use this operation to list the spool files for a batch job on z/OS.

HTTP method and URI path

```
GET /zosmf/restjobs/jobs/<jobname>/<jobid>/files
GET /zosmf/restjobs/jobs/<correlator>/files
```

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job for which the spool files are to be listed. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job for which the spool files are to be listed. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.
- **/files** indicates that the request is to list the spool files for the specified job.

Custom headers

None.

Query parameters

None.

Required authorizations

See “Required authorizations” on page 464.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, note that this request can be directed to a secondary JES subsystem. To do so, use the following URL format:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid/files
```

where *JESB* is the name of the secondary JES subsystem. A request to a secondary JES subsystem must include the job name and job id, rather than a job correlator.

Expected response

On completion, the z/OS jobs REST interface returns an HTTP response with an array of zero or more JSON job file documents. For the contents, see “Job file document” on page 499.

For errors, z/OS jobs REST interface returns an appropriate HTTP status code and error information as a JSON error report document. See “Error report document” on page 500.

Example request

The following request lists the spool files for the job TESTJOB1, job ID JOB00023:

```
GET /zosmf/restjobs/jobs/TESTJOB1/JOB00023/files HTTP/1.1
```

Example response

A sample response is shown in Figure 263.

```
HTTP/1.1 200 OK
Date: Thu, 17 Jan 2013 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

[
{"jobid":"JOB00023","jobname":"TESTJOB1","subsystem":null,"id":1,
"stepname":"JESE","procstep":null,"class":"H",
"ddname":"JESMSGLG","record-count":14,"byte-count":1200,
"records-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB1\/JOB00023\/1/records"},
{"jobid":"JOB00023","jobname":"TESTJOB1","subsystem":null,"id":2,
"stepname":"JESE","procstep":null,"class":"H",
"ddname":"JESJCL","record-count":10,"byte-count":526,
"records-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB1\/JOB00023\/2/records"},
{"jobid":"JOB00023","jobname":"TESTJOB1","subsystem":null,"id":3,
"stepname":"JESE","procstep":null,"class":"H",
"ddname":"JESYSMSG","record-count":14,"byte-count":1255,
"records-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB1\/JOB00023\/3/records"},
{"jobid":"JOB00023","jobname":"TESTJOB1","subsystem":null,"id":4,
"stepname":"STEP57","procstep":"COMPILE","class":"H",
"ddname":"SYSUT1","record-count":6,"byte-count":741,
"records-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB1\/JOB00023\/4/records"},
{"jobid":"JOB00023","jobname":"TESTJOB1","subsystem":null,"id":5,
"stepname":"STEP57","procstep":"COMPILE","class":"A",
"ddname":"SYSPRINT","record-count":3,"byte-count":209,
"records-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOB1\/JOB00023\/5/records"}
]
```

Figure 263. Example: Returned list of spool files

Retrieve the contents of a job spool file

You can use this operation to retrieve the contents of a job spool file on z/OS. Also, you can use this service to retrieve the JCL that was used to submit the job.

HTTP method and URI path

```
GET /zosmf/restjobs/jobs/<jobname>/<jobid>/files/<nnn>/records
GET /zosmf/restjobs/jobs/<correlator>/files/<nnn>/records
GET /zosmf/restjobs/jobs/<jobname>/<jobid>/files/JCL/records
GET /zosmf/restjobs/jobs/<correlator>/files/JCL/records
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job to be used for the request. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job to be used for the request. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.
- **/files<*nnn*>/records** indicates that the request is to retrieve the contents of a job spool file for the specified job. The <*nnn*> parameter is the ID for the spool file from which the contents are to be retrieved.
- **/files/JCL/records** indicates that the request is to retrieve the JCL for the specified job.

Custom headers

You can include the following optional custom HTTP header with this request:

X-IBM-Record-Range

Use this header to retrieve a range of records from a spool file. You can specify this range using either of the following formats:

SSS-EEE

where *SSS* identifies the start record and *EEE* identifies the end record to be retrieved. Both values are relative offsets (0-based). When *EEE* is set to 0, records through the end of the spool file are retrieved.

SSS,NNN

where *SSS* identifies the start record and *NNN* identifies the number of records to be retrieved.

For an example of how this custom header is used, see “Examples” on page 474.

Query parameters

You can specify translation for the returned data through the mode parameter. The following values are valid for mode:

text

The z/OS jobs REST interface translates records from the server codepage to the client codepage and returns the records with Content-Type: plain/text. Trailing spaces are removed and newline characters are used as record separators. This value is the default if you omit the mode parameter.

binary

The z/OS jobs REST interface performs no translation and returns the records with Content-Type: application/octet-stream.

record
The z/OS jobs REST interface performs no translation and returns the records with Content-Type: application/octet-stream. The z/OS jobs REST interface prefixes each record with a 4-byte (big endian) length.

Specifying the mode parameter with any other value, or no value, results in the default: `mode=text`.

Required authorizations

See "Required authorizations" on page 464.

In addition, your user ID requires READ access to the JESSPOOL profile for the spool data set. If no profile exists, only the user who created the spool data set can access, modify, or delete it. For information about spool data set security considerations, see *z/OS JES Application Programming*.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, observe the following considerations for this request:

- The response does not include the Content-Length header. Because the server streams the data rather than buffering it in memory, it is usually not possible for the server to determine the total content length of the data before completing the transfer. For similar reasons, the response does not include the Content-Range header, either.
- This request can be directed to a secondary JES subsystem. To do so, use the following URL format:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid/filesJCL/records
```

where *JESB* is the name of the secondary JES subsystem. A request to a secondary JES subsystem must include the job name and job id, rather than a job correlator.

Expected response

On completion, the z/OS jobs REST interface returns an HTTP response with content type defined by the mode query parameter.

For errors, z/OS jobs REST interface returns an appropriate HTTP status code and error information as a JSON error report document. See "Error report document" on page 500.

Examples

In the following example, the GET method is used to retrieve the contents of spool file 1 for the job TESTJOBJ, job ID JOB00023:

```
GET /zosmf/restjobs/jobs/TESTJOBJ/JOB00023/files/1/records HTTP/1.1
Host: zosmf1.yourco.com
```

A sample response is shown in Figure 264 on page 475.

```
HTTP/1.1 200 OK

Date: Thu, 17 Jan 2013 05:39:28 +0000GMT
Content-Type: text/plain
Connection: close

                J E S 2 J O B L O G -- S Y S T E M E I M G  -- N O D E  D C E I M G W V

15.49.11 JOB00239 ---- MONDAY,    14 JAN 2013 ----
15.49.11 JOB00239  IRR010I  USERID IBMUSER  IS ASSIGNED TO THIS JOB.
15.49.11 JOB00239  ICH70001I IBMUSER  LAST ACCESS AT 15:48:25 ON MONDAY, JANUARY 14, 2013
15.49.11 JOB00239  $HASP373 INSTALL  STARTED - INIT 2    - CLASS A - SYS EIMG
15.49.11 JOB00239  IEF403I INSTALL - STARTED - TIME=15.49.11
15.49.16 JOB00239  IEF404I INSTALL - ENDED - TIME=15.49.16
15.49.16 JOB00239  $HASP395 INSTALL  ENDED
------ JES2 JOB STATISTICS ------
  14 JAN 2013 JOB EXECUTION DATE
           71 CARDS READ
          287 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
           13 SYSOUT SPOOL KBYTES
         0.08 MINUTES EXECUTION TIME
```

Figure 264. Example: Returned spool file content

In the following example, the GET method is used to retrieve a range of records (the first 250) using the X-IBM-Record-Range custom header:

```
GET /zosmf/restjobs/jobs/TESTJOBJ/JOB00023/files/8/records HTTP/1.1
X-IBM-Record-Range: 0-249
```

A sample response is shown in Figure 265.

```
HTTP/1.1 200 OK

Date: Thu, 17 Jan 2013 05:39:28 +0000GMT
Content-Type: text/plain
Connection: close

...(the first 250 records)
```

Figure 265. Example: Returned spool file content (a range of records)

In the following example, the GET method is used to retrieve the JCL for the job TESTJOBJ, job ID JOB00060:

```
GET /zosmf/restjobs/jobs/TESTJOBJ/JOB00060/files/JCL/records HTTP/1.1
```

A sample response is shown in Figure 266.

```
HTTP/1.1 200 OK

//TESTJOBJ JOB (),MSGCLASS=H
// EXEC PGM=IEFBR14
```

Figure 266. Example: Returned job content (the job JCL)

Submit a job

You can use this operation to submit a job to run on z/OS.

HTTP method and URI path

```
PUT /zosmf/restjobs/jobs[/-<JESB>]
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<JESB>*** represents an optionally-specified secondary JES subsystem. If you omit this value, the request is processed by the primary JES subsystem.

Standard headers

Use the following standard HTTP header with this request:

`Content-Type`
One of the following values, as appropriate:

- Set to `text/plain` when the optional header X-IBM-Intrdr-Mode is set to TEXT or is omitted, and the job JCL is included in the request.
- Set to `application/octet-stream` when optional header X-IBM-Intrdr-Mode is set to RECORD or BINARY, and the JCL for the job to be submitted is included in the HTTP request.
- Set to `application/json` when the job to be submitted resides in a data set or UNIX file, which is identified in a JSON document (included as input with this request).

Custom headers

Optionally, you can include one of the following custom HTTP headers with this request:

`X-IBM-Intrdr-Class`
A single character that specifies the internal reader class; the default is A. This value defines the default message class (MSGCLASS) for the job.

`X-IBM-Intrdr-Recfm`
A single character that specifies the internal reader record format: F or V.

When submitting a job from a data set, you can omit this header. Otherwise, this value must match the record format of the data set.

When not submitting a job from a data set, if you omit this header or specify a value other than F or V, the default of F is used.

`X-IBM-Intrdr-Lrecl`
An integer value specifying the internal reader logical record length (LRECL).

When submitting a job from a data set, you can omit this header. Otherwise, this value must match the LRECL of the data set.

When not submitting a job from a data set, if you omit this header or specify a non-integer value, the default of 80 is used.

`X-IBM-Intrdr-Mode`
A keyword that specifies the format of the input job: TEXT, RECORD, or BINARY.

When submitting a job from a data set, you can omit this header. Otherwise, this value must be set to RECORD.

When not submitting a job from a data set, observe the following rules:

- If you omit this header, the TEXT keyword is used by default.
- If you specify the BINARY keyword, the X-IBM-Intrdr-Recfm header must be omitted or set to F (the default).
- If you specify the RECORD keyword or BINARY keyword, you must set Content-Type to `application/octet-stream`.

X-IBM-User-Correlator

Specifies the user portion of the job correlator. In z/OS, a job correlator can be used to associate each job with a unique 64-character value, which provides you with a means to query a job in the system and track it through execution.

A job correlator consists of a 31-byte system-defined portion and a colon character (:), followed by a 32-byte user portion. The system-defined portion contains the following values:

- 8-byte job ID
- 8-byte MAS name for the system on which the job resides
- 8-byte sequence value
- 7-bytes of reserved space.

Following the system value is the colon character (:) separator and the second string: an optional 32-character user-defined value (the user portion). This value is 1 – 32 characters in length, where the first character must be uppercase alphabetic (A-Z) or special ($, #, @). The remaining characters (up to 31) can be any combination of uppercase alphabetic, numeric (0-9), or special. Blank characters are not supported.

If specified, the user portion is combined with the system portion, producing the full job correlator that will be returned in the job-correlator property of the JSON job document. If the user portion is not specified, the returned job correlator is the 32-byte system value, ending with the colon (:).

If this header is specified when JES3 is the primary job entry subsystem, an error will result and no job is submitted.

For more information on the job correlator, see *z/OS JES2 Commands*.

X-IBM-JCL-Symbol-*name*

Specifies the name and value for a JCL symbol. The symbol name is included in the header, at the name position. The characters following 'X-IBM-JCL-Symbol-' up to the colon separator (:) form the symbol name. The data following the colon specifies the value for the symbol.

A symbol name is 1 – 8 characters, where the first character must be uppercase alphabetic (A-Z) or special ($, #, @). The remaining characters (up to 7) can be any combination of uppercase alphabetic, numeric (0-9), or special.

A symbol value is limited to 255 characters. Multiple symbol names and values can be specified, up to a limit of 128.

Example: `X-IBM-JCL-Symbol-MBR: ABC` specifies symbol name `MBR` with value `ABC`. Specifying this custom header and submitting a job that uses `//MYDD DSN=MY.DATASET(&MBR.),DISP=SHR` in the JCL will cause `ABC` to be substituted as the member name.

If this header is specified when JES3 is the primary job entry subsystem, an error will result and no job will be submitted.

For more information on JCL symbols, see *z/OS MVS JCL Reference*.

X-IBM-Notification-URL

Specifies a destination URL for receiving an HTTP POST when the job is no longer eligible for execution (that is, when the job reaches the output queue or purge queue). The notification is in the form of a JSON document (`Content-Type: application/json`), which contains job status information. For the contents of the JSON document, see "Job completed document" on page 497.

Query parameters

None.

Input to this request

- Internet media type: text/plain, application/octet-stream, or application/json
- HTTP request with optional headers, followed by job to be submitted or a JSON document identifying the location of the job to be submitted (a data set or UNIX file).

To submit a job, you can include the job JCL in the HTTP request itself, or you can have the request refer to a job that resides in a data set or UNIX file. Here, you include a JSON document (`"Content-Type: application/json"` with the HTTP request. The JSON document contains the property `"file":"<file-name>"` where *<file-name>* identifies the data set or UNIX file that contains the job to be submitted.

Use the JSON document to identify the data set or UNIX file containing the job to be submitted, as follows:

- For a data set, specify the qualified data set name, prefixing the data set name with two leading forward slash characters ("//").

  If not fully qualified, the current z/OSMF user ID is prefixed to the data set name. Supported data set types include sequential data sets and members of partitioned data sets.

  Data sets must be catalogued.
- For a z/OS UNIX file, specify the absolute path name of the file.

  Codepage conversion is not performed on the contents of the file.

For a migrated data set, this operation does not cause the data set to be retrieved, unless you specify otherwise. To request that a data set be recalled without waiting, you can specify the "recall" property with the value "yesnowait" to the input JSON document. Unique error responses are provided when a migrated data set is requested to be recalled without waiting and for when a migrated data set is not requested to be recalled. In both cases, no job is submitted. If you have asked for a recall, without waiting, when you try the submit again, you should do so without adding the "recall" property to the JSON document or by changing the "recall" property to the value "no."

Required authorizations

See "Required authorizations" on page 464.

In addition, your user ID must be authorized to run jobs on the system and be able to access any protected resources that the job might require. For information about the security considerations for job submission, see *z/OS JES2 Initialization and Tuning Guide*, SA22-7532, or *z/OS JES3 Initialization and Tuning Guide*, SA22-7549.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, observe the following considerations for this request:

- This request can be directed to a secondary JES subsystem. To do so, use the following request format:

```
https://host:port/zosmf/restjobs/jobs/-JESB
```

  where *JESB* is the name of the secondary JES subsystem in these examples.

Expected response

On completion, the z/OS jobs REST interface returns an HTTP response with a JSON job document. For the contents, see "Job document" on page 496.

For errors, z/OS jobs REST interface returns an appropriate HTTP status code and error information as a JSON error report document. See "Error report document" on page 500.

Example of submitting a job from a data set or UNIX file

Table 257 shows variations of a PUT request that submits the job TESTJOBX to run on z/OS. In each variation, the PUT request contains a JSON statement that identifies the location of the job to be submitted.

Table 257. Variations of a PUT request for submitting a job from a data set or UNIX file.

Location of the job	Example

Partitioned data set (fully qualified)	`PUT /zosmf/restjobs/jobs HTTP/1.1`
	`Content-Type: application/json`
	`X-IBM-Intrdr-Class: A`
	
	`{ "file" : "//'MYJOBS.TEST.CNTL(TESTJOBX)'" }`

Partitioned data set (non-fully qualified)	`PUT /zosmf/restjobs/jobs HTTP/1.1`
	`Content-Type: application/json`
	`X-IBM-Intrdr-Class: A`
	
	`{ "file" : "//TEST.CNTL(TESTJOBX)" }`

Sequential data set	`PUT /zosmf/restjobs/jobs HTTP/1.1`
	`Content-Type: application/json`
	`X-IBM-Intrdr-Class: A`
	
	`{ "file" : "//'MYJOBS.TEST.JOB1'" }`

UNIX file	`PUT /zosmf/restjobs/jobs HTTP/1.1`
	`Content-Type: application/json`
	`X-IBM-Intrdr-Class: A`
	`X-IBM-Intrdr-Recfm: V`
	`X-IBM-Intrdr-Lrecl: 255`
	`X-IBM-Intrdr-Mode: TEXT`
	
	`{ "file" : "/u/myjobs/job1" }`

Example of a request that contains the job JCL

The following request submits the job TESTJOBX to run on z/OS. Here, the JCL for the job to be submitted is contained in the PUT request.

```
PUT /zosmf/restjobs/jobs HTTP/1.1
Host: zosmf1.yourco.com
Content-Type: text/plain
X-IBM-Intrdr-Class: A
X-IBM-Intrdr-Recfm: F
X-IBM-Intrdr-Lrecl: 80
X-IBM-Intrdr-Mode: TEXT

//TESTJOBX JOB (),MSGCLASS=H
// EXEC PGM=IEFBR14
```

A sample response is shown in Figure 267.

```
HTTP/1.1 201 Created
Date: Fri, 17 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
"jobid":"JOB00025","jobname":"TESTJOBX","subsystem":null,"owner":"IBMUSER",
"status":"INPUT","type":"JOB","class":"A","retcode":null,
"url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOBX\/JOB00025",
"files-url":"https:\/\/host:port\/zosmf\/restjobs\/jobs\/TESTJOBX\/JOB00025\/files"
}
```

Figure 267. Example: Returned results of a job submission

Hold a job

For a job that has been submitted to run on z/OS, but not yet selected for processing, you can use this operation to hold the job. When held, a job is not be eligible for selection.

You can use a similar method to release the job and make is available for selection; see “Release a job” on page 484.

HTTP method and URI path

```
PUT /zosmf/restjobs/jobs/<jobname>/<jobid>
PUT /zosmf/restjobs/jobs/<correlator>
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job to be held. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job to be held. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.

Custom headers

None.

Query parameters

None.

Input to this request

- Internet media type: application/json
- HTTP request with JSON document containing the following properties:

  “request”:”hold”
  Indicates a request to hold a job.

  “version”:”*n.n*”
  Specifies the version of the service to be used, either 1.0 or 2.0.

To request asynchronous processing for this service (the default), set the "version" property to `1.0` or omit the property from the request. To request synchronous processing, set "version" to `2.0`. If so, the system will attempt to process the request synchronously, if such processing is supported on the target JES subsystem.

For further considerations, see “Synchronous support for the job modify operations” on page 463.

Required authorizations

See “Required authorizations” on page 464.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, observe the following consideration for this request:

- This request can be directed to a secondary JES subsystem. To do so, use one of the following request formats:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid
https://host:port/zosmf/restjobs/jobs/-JESB/correlator
```

  where:
  - *JESB* is the name of the secondary JES subsystem.
  - The job to be held is identified by either the job name and job ID (*jobname/jobid*) or the job correlator (*correlator*).

Expected response

The response depends on whether the request is processed synchronously or asynchronously, as follows:

- For an asynchronous request, the caller receives only the HTTP status code `202 ACCEPTED`. To determine whether the request was successful, the caller can issue the service described in “Obtain the status of a job” on page 466.
- For a synchronous request, the caller receives an HTTP status code, which indicates the results of the request, as follows:
  - Status code 200 indicates that the synchronous request was processed successfully. This status, however, does not mean that the operation was successful. To determine the success of the operation, check the "status" property in the JSON job feedback document for a value of 0 (zero). See “Job feedback document” on page 498.
  - Status code of 4*nn* or 5*nn* indicates that an HTTP error has occurred.

For errors, z/OS jobs REST interface returns error information as a JSON error report document. See “Error report document” on page 500.

Example request

The following request specifies that the job TESTJOBW, job ID JOB00023, is to be held. With the inclusion of the "version" property set to `2.0`, the request is eligible to be processed synchronously, if supported on the target JES subsystem.

```
PUT /zosmf/restjobs/jobs/TESTJOBW/JOB00023 HTTP/1.1
Host: zosmf1.yourco.com
Content-Length: 40
Content-Type: application/json

{
"request":"hold",
"version":"2.0"
}
```

Example response

A sample response is shown in Figure 268.

```
HTTP/1.1 200 OK
Date: Thu, 16 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
"jobid":"JOB00023",
"jobname":"TESTJOBW",
"original-jobid":"JOB00023",
"owner":"IBMUSER"
"member":"JES2",
"sysname":"SY1",
"job-correlator":"J0000023SY1.....CC20F378.......:",
"status":"0"
}
```

Figure 268. Example: Returned results of a job hold request

Release a job

For a job that has been held from execution on z/OS, you can use this operation to release the job. When released, a job is made eligible for selection to execute.

HTTP method and URI path

```
PUT /zosmf/restjobs/jobs/<jobname>/<jobid>
PUT /zosmf/restjobs/jobs/<correlator>
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job to be released. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job to be released. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.

Custom headers

None.

Query parameters

None.

Input to this request

- Internet media type: application/json
- HTTP request with JSON document containing the following properties:

  "request":"release"
  Indicates a request to release a job.

  "version":"*n.n*"
  Specifies the version of the service to be used, either 1.0 or 2.0.

To request asynchronous processing for this service (the default), set the "version" property to `1.0` or omit the property from the request. To request synchronous processing, set "version" to `2.0`. If so, the system will attempt to process the request synchronously, if such processing is supported on the target JES subsystem.

For further considerations, see "Synchronous support for the job modify operations" on page 463.

Required authorizations

See "Required authorizations" on page 464.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, observe the following consideration for this request:

- This request can be directed to a secondary JES subsystem. To do so, use one of the following request formats:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid
https://host:port/zosmf/restjobs/jobs/-JESB/correlator
```

  where:
  - *JESB* is the name of the secondary JES subsystem.
  - The job to be released is identified by either the job name and job ID (*jobname*/*jobid*) or the job correlator (*correlator*).

Expected response

The response depends on whether the request is processed synchronously or asynchronously, as follows:

- For an asynchronous request, the caller receives only the HTTP status code `202 ACCEPTED`. To determine whether the request was successful, the caller can issue the service described in "Obtain the status of a job" on page 466.
- For a synchronous request, the caller receives an HTTP status code, which indicates the results of the request, as follows:
  - Status code 200 indicates that the synchronous request was processed successfully. This status, however, does not mean that the operation was successful. To determine the success of the operation, check the "status" property in the JSON job feedback document for a value of 0 (zero). See "Job feedback document" on page 498.
  - Status code of 4*nn* or 5*nn* indicates that an HTTP error has occurred.

For errors, z/OS jobs REST interface returns error information as a JSON error report document. See "Error report document" on page 500.

Example request

The following request specifies that the job TESTJOBW, job ID JOB00023, is to be released. With the inclusion of the "version" property set to `2.0`, the request is eligible to be processed synchronously, if supported on the target JES subsystem.

```
PUT /zosmf/restjobs/jobs/TESTJOBW/JOB00023 HTTP/1.1
Host: zosmf1.yourco.com
Content-Length: 40
Content-Type: application/json

{
"request":"release"
"version":"2.0"
}
```

Example response

A sample response is shown in Figure 269 on page 486.

```
HTTP/1.1 200 OK
Date: Thu, 16 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
"jobid":"JOB00023",
"jobname":"TESTJOBW",
"original-jobid":"JOB00023",
"owner":"IBMUSER"
"member":"JES2",
"sysname":"SY1",
"job-correlator":"J0000023SY1.....CC20F378.......:",
"status":"0"
}
```

Figure 269. Example: Returned results of a job release request

Change the job class

You can use this operation to change the class of a job on z/OS.

HTTP method and URI path

```
PUT /zosmf/restjobs/jobs/<jobname>/<jobid>
PUT /zosmf/restjobs/jobs/<correlator>
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job for which the class is to be changed. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job for which the class is to be changed. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.

Custom headers

None.

Query parameters

None.

Input to this request

- Internet media type: application/json
- HTTP request with JSON document containing the following properties:

  "class":"*<new_job_class>*"
  : Indicates a request to change the job class to the value *<new_job_class>*.

  "version":"*n.n*"
  : Specifies the version of the service to be used, either 1.0 or 2.0.

To request asynchronous processing for this service (the default), set the "version" property to `1.0` or omit the property from the request. To request synchronous processing, set "version" to `2.0`. If so, the system will attempt to process the request synchronously, if such processing is supported on the target JES2 subsystem.

For further considerations, see "Synchronous support for the job modify operations" on page 463.

Required authorizations

See "Required authorizations" on page 464.

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

In addition, observe the following considerations for this request:

- The specified job class is not validated on input. To verify the success of this request, your program can issue a GET request for the job status, and check the class value in the returned JSON Job document. See "Obtain the status of a job" on page 466.
- This request can be directed to a secondary JES subsystem. To do so, use the following request format:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid
```

  where *JESB* is the name of the secondary JES subsystem.
- A request to a secondary JES subsystem must include the job name and job id, rather than a job correlator.

Expected response

The response depends on whether the request is processed synchronously or asynchronously, as follows:

- For an asynchronous request, the caller receives only the HTTP status code `202 ACCEPTED`. To determine whether the request was successful, the caller can issue the service described in "Obtain the status of a job" on page 466.
- For a synchronous request, the caller receives an HTTP status code, which indicates the results of the request, as follows:
  - Status code 200 indicates that the synchronous request was processed successfully. This status, however, does not mean that the operation was successful. To determine the success of the operation, check the "status" property in the JSON job feedback document for a value of 0 (zero). See "Job feedback document" on page 498.
  - Status code of 4*nn* or 5*nn* indicates that an HTTP error has occurred.

For errors, z/OS jobs REST interface returns error information as a JSON error report document. See "Error report document" on page 500.

Example request

The following request specifies job class A for the job TESTJOBW, job ID JOB00023. With the inclusion of the "version" property set to `2.0`, the request is eligible to be processed synchronously, if supported on the target JES subsystem.

```
PUT /zosmf/restjobs/jobs/TESTJOBW/JOB00023 HTTP/1.1
Host: zosmf1.yourco.com
Content-Length: 40
Content-Type: application/json

{
"class":"A"
"version":"2.0"
}
```

Example response

A sample response is shown in Figure 270 on page 489.

```
HTTP/1.1 200 OK
Date: Thu, 16 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
"jobid":"JOB00023",
"jobname":"TESTJOBW",
"original-jobid":"JOB00023",
"owner":"IBMUSER"
"member":"JES2",
"sysname":"SY1",
"job-correlator":"J0000023SY1.....CC20F378.......:",
"status":"0"
}
```

Figure 270. Example: Returned results of a job class change

Cancel a job

You can use this operation to cancel a job on z/OS.

HTTP method and URI path

```
PUT /zosmf/restjobs/jobs/<jobname>/<jobid>
PUT /zosmf/restjobs/jobs/<correlator>
```

---

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job to be canceled. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job to be canceled. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.

Custom headers

None.

Query parameters

None.

Input to this request

- Internet media type: application/json
- HTTP request with JSON document containing the following properties:

  "request":"cancel"
  : Indicates a request to cancel a job.

  "version":"*n.n*"
  : Specifies the version of the service to be used, either 1.0 or 2.0.

To request asynchronous processing for this service (the default), set the "version" property to `1.0` or omit the property from the request. To request synchronous processing, set "version" to `2.0`. If so, the system will attempt to process the request synchronously, if such processing is supported on the target JES2 subsystem.

For further considerations, see "Synchronous support for the job modify operations" on page 463.

Required authorizations

See "Required authorizations" on page 464.

In addition, your user ID must be authorized to cancel the job on the system. For information about the security considerations for job cancellation, see *z/OS JES2 Initialization and Tuning Guide*, SA22-7532, or *z/OS JES3 Initialization and Tuning Guide*, SA22-7549.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, note that this request can be directed to a secondary JES subsystem. To do so, use the following URL format:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid
```

where *JESB* is the name of the secondary JES subsystem in these examples. A request to a secondary JES subsystem must include the job name and job id, rather than a job correlator.

Expected response

The response depends on whether the request is processed synchronously or asynchronously, as follows:

- For an asynchronous request, the caller receives only the HTTP status code `202 ACCEPTED`. To determine whether the request was successful, the caller can issue the service described in “Obtain the status of a job” on page 466.
- For a synchronous request, the caller receives an HTTP status code, which indicates the results of the request, as follows:
  - Status code 200 indicates that the synchronous request was processed successfully. This status, however, does not mean that the operation was successful. To determine the success of the operation, check the "status" property in the JSON job feedback document for a value of 0 (zero). See “Job feedback document” on page 498.
  - Status code of 4*nn* or 5*nn* indicates that an HTTP error has occurred.

For errors, z/OS jobs REST interface returns error information as a JSON error report document. See “Error report document” on page 500.

Example request

The following request cancels the job TESTJOB2, job ID JOB00084 on z/OS. To request synchronous processing by the target JES subsystem, the request includes the specification `"version":"2.0"`.

```
PUT /zosmf/restjobs/jobs/TESTJOB2/JOB00084 HTTP/1.1
Host: zosmf1.yourco.com
Content-Length: 40
Content-Type: application/json

{
"request":"cancel",
"version":"2.0"
}
```

Example response

A sample response is shown in Figure 271 on page 492.

```
HTTP/1.1 200 OK
Date: Thu, 16 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
"jobid":"JOB00084",
"jobname":"TESTJOB2",
"original-jobid":"JOB00084",
"owner":"IBMUSER"
"member":"JES2",
"sysname":"SY1",
"job-correlator":"J0000084SY1.....CC20F378.......:",
"status":"0"
}
```

Figure 271. Example: Returned results of a job cancellation

Cancel a job and purge its output

You can use this operation to cancel a job and purge its output.

HTTP method and URI path

```
DELETE /zosmf/restjobs/jobs/<jobname>/<jobid>
DELETE /zosmf/restjobs/jobs/<correlator>
```

where:

- **/zosmf/restjobs/jobs/** identifies the z/OS jobs REST interface.
- ***<jobname>/<jobid>*** identifies the job to be canceled. Use either the job name and job ID combination or the job correlator to identify the job.
- ***<correlator>*** identifies the job to be canceled. Use either the job name and job ID combination or the job correlator to identify the job.

  To use a job correlator on this request, specify the full job correlator for the job: The 31-byte system portion, a semicolon, and the user portion (up to 32 bytes). The correlator can be one that you have obtained from the "job-correlator" property in a returned JSON job document. Alternatively, you can specify the complete URL as provided in the "url" property of a JSON job document.

Custom headers

You can include the following optional custom HTTP header with this request:

X-IBM-Job-Modify-Version

Use this header to specify whether the request is to be processed asynchronously or synchronously, as follows:

1.0 Specifies that the request is to be processed asynchronously. In response, the caller receives an HTTP status code of `202 Accepted`, with no indication of the success or failure of the request. To verify that the initial request was performed, the caller can issue the service described in "Obtain the status of a job" on page 466.

2.0 Specifies that the request is to be processed synchronously. In response, the caller receives an HTTP status code, which indicates the results of the request. For a successful request, the caller also receives the JSON job feedback document, which includes details about the job that was cancelled.

If this header is omitted, the request is processed asynchronously by default.

Synchronous processing is supported for JES2 only. On systems running JES3, the z/OS jobs REST interface services must run asynchronously.

For an example of how this header is specified, see "Example request" on page 494.

Query parameters

None.

Required authorizations

See "Required authorizations" on page 464.

In addition, your user ID must be authorized to cancel the job on the system, which allows the user to delete the job SYSOUT data sets. For information about the security considerations for job cancellation, see *z/OS JES2 Initialization and Tuning Guide*, SA22-7532, or *z/OS JES3 Initialization and Tuning Guide*, SA22-7549.

Usage considerations

See “Usage considerations for the z/OSMF REST services” on page 2.

In addition, note that this request can be directed to a secondary JES subsystem. To do so, use the following URL format:

```
https://host:port/zosmf/restjobs/jobs/-JESB/jobname/jobid
```

where *JESB* is the name of the secondary JES subsystem. A request to a secondary JES subsystem must include the job name and job id, rather than a job correlator.

Expected response

The response depends on whether the request is processed synchronously or asynchronously, as follows:

- For an asynchronous request, the caller receives only the HTTP status code `202 ACCEPTED`. To determine whether the request was successful, the caller can issue the service described in “Obtain the status of a job” on page 466.
- For a synchronous request, the caller receives an HTTP status code, which indicates the results of the request, as follows:
  - Status code 200 indicates that the synchronous request was processed successfully. This status, however, does not mean that the operation was successful. To determine the success of the operation, check the "status" property in the JSON job feedback document for a value of 0 (zero). See “Job feedback document” on page 498.
  - Status code of 4*nn* or 5*nn* indicates that an HTTP error has occurred.

For HTTP errors, z/OS jobs REST interface returns error information as a JSON error report document. See “Error report document” on page 500.

Example request

The following request cancels the job TESTJOBW, job ID JOB00085 and purges its output on the z/OS system. With the inclusion of the **`X-IBM-Job-Modify-Version`** header set to `2.0`, the request is eligible to be processed synchronously, if supported on the target JES subsystem.

```
DELETE /zosmf/restjobs/jobs/TESTJOBW/JOB00085 HTTP/1.1
X-IBM-Job-Modify-Version: 2.0
```

Example response

A sample response is shown in Figure 272 on page 495. Because the request was processed synchronously by the target JES subsystem, the response body includes the job feedback document with details about the job that was cancelled.

```
HTTP/1.1 200 OK
Date: Thu, 16 Jan 2014 05:39:28 +0000GMT
Content-Type: application/json
Connection: close

{
"jobid":"JOB00085",
"jobname":"TESTJOBW",
"original-jobid":"JOB00085",
"owner":"IBMUSER"
"member":"JES2",
"sysname":"SY1",
"job-correlator":"J0000085SY1.....CC20F380.......:",
"status":"0"
}
```

Figure 272. Example: Results of a job delete request

JSON document specifications for z/OS jobs REST interface requests

This section describes the contents of the JSON documents that are used with z/OS jobs REST interface requests.

The following JSON documents are described:
- "Job document"
- "Job completed document" on page 497
- "Job feedback document" on page 498
- "Job file document" on page 499
- "Error report document" on page 500.

Job document

Table 258 shows the contents of the JSON job document.

Table 258. Contents of the JSON job document

Property	Description
jobid	Job ID.
jobname	Job name.
subsystem	The primary or secondary JES subsystem. If this value is null, the job was processed by the primary subsystem.
owner	The z/OS user ID associated with the job.

status={
INPUT |
ACTIVE |
OUTPUT
}
	Job status. One of the following values:
	INPUT
	Job is in input processing.
	ACTIVE
	Job is running.
	OUTPUT
	Job is on the hardcopy output queue.
	
	If this value is null, the job status could not be determined.

type={
JOB |
STC |
TSU
}
	Job type. One of the following values:
	JOB
	Batch job.
	STC
	Started task.
	TSU
	TSO/E user.

class	Job execution class.

Table 258. Contents of the JSON job document (continued)

Property	Description

retcode={
ABENDU*nnnn* |
ABEND S*xxx* |
CANCELED |
CC nnnn |
CONV ABEND |
CONV ERROR |
JCL ERROR |
SEC ERROR |
SYS FAIL
}
	Job completion code. One of the following values:
	ABENDU*nnnn*
	Job ended with the user abend code *nnnn*.
	ABEND S*xxx*
	Job ended with the system abend code *xxx*.
	CANCELED
	Job was canceled.
	CC *nnnn*
	Job ended with the completion code *nnnn*.
	CONV ABEND
	Converter ended abnormally when processing the job.
	CONV ERROR
	Converter error when processing the job.
	JCL ERROR
	Job encountered a JCL error.
	SEC ERROR
	Job failed a security check.
	SYS FAIL
	System failure.
	
	If this value is null, the job has not yet completed.

url	Resource URL based on original HTTP request.
files-url	Resource URL for listing the spool files for the job.
job-correlator	Job correlator. If this value is null, the job was submitted to JES3.
phase	Job phase. Provides a numeric indicator of the current state of the job.
phase-name	Job phase name. Provides a text description of the specific phase of the job.
reason-not-running	Text identifying one or more reasons why the job is not running.

Job completed document

Table 259 shows the contents of the JSON job completed document.

Table 259. Contents of the JSON job completed document

Property	Description
job-correlator	Job correlator. If this value is null, the job was submitted to JES3.
jobid	Job ID.
jobname	Job name.
owner	The z/OS user ID associated with the job.
class	Job execution class.

Table 259. Contents of the JSON job completed document (continued)

Property	Description

retcode={
ABENDU*nnnn* |
ABEND S*xxx* |
CANCELED |
CC nnnn |
CONV ABEND |
CONV ERROR |
JCL ERROR |
SEC ERROR |
SYS FAIL
}
	Job completion code. One of the following values:
	ABENDU*nnnn*
	Job ended with the user abend code *nnnn*.
	ABEND S*xxx*
	Job ended with the system abend code *xxx*.
	CANCELED
	Job was canceled.
	CC *nnnn*
	Job ended with the completion code *nnnn*.
	CONV ABEND
	Converter ended abnormally when processing the job.
	CONV ERROR
	Converter error when processing the job.
	JCL ERROR
	Job encountered a JCL error.
	SEC ERROR
	Job failed a security check.
	SYS FAIL
	System failure.

completion-type	Specific completion type: **0** No completion information was received. **1** Job ended normally. **2** Job ended with a completion code. **3** Job encountered a JCL error. **4** Job was canceled. **5** Job abended. **6** Converter error when processing the job. **7** Job encountered a security error. **8** Job failed in EOM. **9** Job failed a security check. **10** System failure.
completion-code	Completion code. Set for completion-type values 1 and 2. Otherwise null.
abend-code	Job completed with abend code. Set for completion-type values 5 and 8. Otherwise null. When set, one of the following values: **U***nnnn* Job ended with the user abend code *nnnn*. **S***xxx* Job ended with the system abend code *xxx*.

Job feedback document

Table 260 shows the contents of the JSON job feedback document.

Table 260. Contents of the JSON job feedback document

Property	Description
jobid	Job ID.
jobname	Job name.
original-jobid	Original job ID. If the job was processed on another system, this value represents the original job identifier that was assigned when the job was submitted on the host system. If the target system cannot assign the original job identifier, the target system assigns a new ID to the job, which is indicated as "jobid" in this document.
owner	z/OS user ID associated with the job.

Table 260. Contents of the JSON job feedback document (continued)

Property	Description
member	JES2 multi-access spool (MAS) member name.
sysname	z/OS system name.
job-correlator	Job correlator. If this value is null, the job was submitted to JES3.
status={*n*}	job processing status. If set to zero (0), the request was processed successfully. Otherwise, there was an error. See the message property for a description of the error.
internal-code	If job processing status indicates an error (a value other than 0), this property contains the internal service routine return code. Otherwise, this property is omitted.
message	If job processing status indicates an error (a value other than 0), this property contains a description of the error. Otherwise, this property is omitted.

Job file document

Table 261 shows the contents of the JSON job file document.

Table 261. Contents of the JSON job file document

Property	Description
jobname	Job name.
recfm	Record format of the file. The first character of the returned string is one of the following: **F** Fixed length records **V** Variable length records **U** Undefined length records. One or more subsequent characters might also be present in the returned string (in this order): **B** File has blocked records **S** File has standard records (if fixed length format) or spanned records (if variable length format) **M** File has machine print-control characters **A** File has ASA (ANSI) print-control characters. Generally, the B (blocked) and S (standard or spanned) characters are not present for JES spool files. Also, the M (machine) and A (ASA) characters are mutually exclusive.
byte-count	Number of bytes on spool consumed by the spool file. The value can be zero (0). This field is integer data type.
record-count	Number of records in the spool file. The value can be zero (0). This field is integer data type.
job-correlator	Job correlator. If this value is null, the job was submitted to JES3.
class	Class assigned to the spool file.
jobid	Job ID.
id	Data set number (key). This field is integer data type.
ddname	DDNAME for the data set creation.
records-url	Resource URL for retrieving the spool file contents for the job.
lrecl	Specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records.
subsystem	The primary or secondary JES subsystem. If the value is null, the job was processed by the primary subsystem.
stepname	Step name for the step that created this data set. The value can be null.
procstep	Procedure name for the step that created this data set. The value can be null.

Error report document

Table 262 shows the contents of the JSON error report document.

Table 262. Contents of the JSON error report document

Property	Description
category	Error category. This field is integer data type.
rc	Return code. This field is integer data type.
reason	Reason code. This field is integer data type.
message	Message that describes the error.
details	(optional) Array of strings containing additional message details.
stack	Stack trace of the exception.

For the meanings of the category, rc, and reason fields, see "Error reporting categories" on page 501.

Error reporting categories

This section describes the error categories and associated error codes that can be returned in the JSON error report document, described in "Error report document" on page 500.

Categories

Table 263 shows the error categories that are defined for errors returned in z/OS jobs REST interface operations.

Table 263. Error categories for z/OS jobs REST interface operations

Category	Ordinal Value	Description	Where the error details are described
Dynalloc	1	Dynamic allocation errors.	"Category 1 — Dynamic allocation error" on page 502
VSAM API	3	Errors produced or detected by the Java/ JNI/ C/ HLASM/ VSAM layer.	"Category 3 — VSAM API error" on page 503
VSAM system	4	Errors produced or detected by VSAM. The return code and reason code are VSAM specific.	"Category 4 — VSAM system error" on page 503
VSAM ABEND	5	ABEND information resulting from VSAM failures.	"Category 5 — VSAM ABEND error" on page 503
Service	6	Errors produced or detected in the service layer.	"Category 6 — Service error" on page 503
Unexpected	7	Unexpected errors detected.	"Category 7 — Unexpected error" on page 507
SSI extended status	8	Errors produced or detected by the extended status function call of the subsystem interface (SSI Function Code 80).	"Category 8 — SSI extended status error" on page 507
CIM	9	Errors produced or detected by the CIM interface.	"Category 9 — Common Information Model (CIM) error" on page 507
SSI job modify	10	Errors produced or detected by the job modify function call of the subsystem interface (SSI Function Code 85).	"Category 10 — SSI job modify error" on page 508

Category 1 — Dynamic allocation error

Table 264 shows the possible conditions for this error category.

Table 264. Category 1 errors

rc	reason	message	Description
n	0	`Error allocating internal reader, RC=%d (0x%08X)`	An error occurred when z/OS attempted to allocate the internal reader for job submission. In the message, *RC* is error data from the dynamic allocation request (SVC 99). To diagnose the error, convert the *RC* value from decimal to a 4-byte hexadecimal value, which provides the dynamic allocation error code and info code, as follows: • High-order two bytes indicate the error code from the dynamic allocation request (field S99ERROR in the input request block S99RB). • Low order two bytes indicate the info code from the dynamic allocation request (field S99INFO in the input request block S99RB). For information about dynamic allocation and the meanings of the error code and info code, see *z/OS MVS Programming: Authorized Assembler Services Guide*.
n	1	`Error allocating input data set: %s, RC=%d (0x%08X)`	An error occurred when z/OS attempted to allocate a ddname for the data set specified as the source for the input job. In the message, *RC* is the return code from the BPXWDYN service. For information about BPXWDYN and the meaning of the return code, see *z/OS Using REXX and z/OS UNIX System Services*.
n	2	`Error allocating spool file: job '%s' spool file id %d, RC=%d (0x%08X)`	An error occurred when z/OS attempted to allocate the requested spool file. Perhaps a thread is attempting to allocate the spool file while another thread is requesting that the job be canceled and its output purged. In the message, *RC* is error data from the dynamic allocation request (SVC 99). Both decimal and hexadecimal values are provided in the message. The hexadecimal value provides the dynamic allocation error code and info code, as follows: • High-order two bytes indicate the error code from the dynamic allocation request (field S99ERROR in the input request block S99RB). • Low order two bytes indicate the info code from the dynamic allocation request (field S99INFO in the input request block S99RB). For information about dynamic allocation and the meanings of the error code and info code, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Category 3 — VSAM API error

Table 265 shows the possible conditions for this error category.

Table 265. Category 3 errors

rc	reason	message	Description
4	1	`Incorrect JesVsam handle`	
4	2	`VSAM file is not open`	
4	3	`Record length %d > lrecl %d`	Writing a record to a VSAM file failed because an incorrect record length was specified.
4	4	`Could not write JCL to internal reader`	An I/O exception occurred when writing JCL to the internal reader.
8	0	`JesVsam get failed`	Buffer too small to hold the VSAM record.
255	0	`JesVsam native buffer malloc failed`	

Category 4 — VSAM system error

Table 266 shows the possible conditions for this error category.

Table 266. Category 4 errors

rc	reason	message	Description
n	*m*	varies	For descriptions of the specific return and reason codes, see the VSAM publications.

Category 5 — VSAM ABEND error

Table 267 shows the possible conditions for this error category.

Table 267. Category 5 errors

rc	reason	message	Description
n	*m*	varies	The values *n* and *m* indicate the ABEND return code and reason code.

Category 6 — Service error

Table 268 shows the possible conditions for this error category.

Table 268. Category 6 errors

rc	reason	message	Description
4	1	`Incorrect Internal Reader mode: %s. Must be one of TEXT | RECORD | BINARY`	Request header X-IBM-Intrdr-Mode specified a value that is not valid. Valid values are TEXT, BINARY, or RECORD.
4	2	`Incorrect Internal Reader parameters: %s. Fixed records are required for binary mode`	The internal reader characteristics form a combination that is not valid. If you specify the value BINARY for the X-IBM-Intrdr-Mode request header, you must specify the value F for the X-IBM-Intrdr-Recfm request header.
4	3	`Request does not contain '%s' content`	Job modify requests must have a content type of application/json.

Table 268. Category 6 errors (continued)

rc	reason	message	Description
4	4	`Value of %s query parameter is not valid`	The query parameter identified in the message either contains incorrect characters or exceeds the allowable length. In the message, the query parameter is *owner*, *prefix*, *jobid*, or *job-correlator*.
4	5	`Update request is not 'cancel'`	The job modify request value for the "request" property is not valid. The value must be “cancel.”
4	6	`Request does not contain a valid job update request`	The job modify request input document does not specify a valid property. The valid properties are: • “request” • “class”
4	7	`No match for method %s and pathInfo='%s'`	The supplied servlet pathinfo does not match any expected string for the HTTP method that was specified.
4	8	`POST requests not supported`	For standard REST requests, the POST HTTP method is not allowed. To avoid this message, include the X-IBM-Requested-Method header to have the request sent through the POST verb.
4	9	`Job submission error. Record length %d too long for JCL submission, maxlen=%d`	The check for record mode job submission failed.
4	10	`No job found for reference: '%s'`	The job modify request specified a job that does not exist.
4	11	`Record range '%s' is not valid for spool file record request`	Request header X-IBM-Record-Range specified a value that is not valid. The content range must be specified using one the following formats: **SSS-EEE** where *SSS* identifies the start record and *EEE* identifies the end record to be retrieved. Both values are relative offsets (0-based). When *EEE* is set to 0, records through the end of the spool file are retrieved. **SSS,NNN** where *SSS* identifies the start record and *NNN* identifies the number of records to be retrieved.
4	12	`Job '%s' does not contain spool file id %d`	
4	13	`Job input was not recognized by system as a job`	The job was submitted without a job statement or with unrecognized (non-JCL) content.
4	14	`Unsupported encoding: %s`	When submitting a job, the Content-Type request header specified a charset value that is not supported.
4	15	`DD names are not supported for submit input`	The value of the “filename” property in the JSON document provided on the submit job interface started with `//DD:` indicating the `dd:ddname` syntax. This is not supported.
4	16	`Data set not found`	The data set specified in the JSON document provided on the submit job interface was not found. It is possible the data set is not catalogued.
4	17	`Submit input data does not start with a slash`	This error occurs when the first character of the input job is not the EBCDIC slash character. Possible causes include: • The Content-Type request header is set to `text/plain` when a JSON document naming the source of the input job is also used. • The input data set or file does not contain EBCDIC data.

Table 268. Category 6 errors (continued)

rc	reason	message	Description
4	18	`Submit input filename must be absolute path: %s`	The z/OS UNIX file specification in the JSON document was not an absolute path.
4	19	`Internal reader mode must be RECORD for data set submission: %s`	If specified, the internal reader mode must be set to RECORD when submitting a job from a data set.
4	20	`Service not implemented: %s`	The requested service has not been implemented. The variable text *%s* contains additional information.
4	22	`Internal reader RECFM (%s) does not match data set RECFM (%s): %s`	If specified, the internal reader record format must match the record format of the existing data set when submitting a job from a data set.
4	23	`Internal reader LRECL (%d) does not match data set LRECL (%d): %s`	If specified, the internal reader logical record length must match the logical record length of the existing data set when submitting a job from a data set.
4	24	`Content type '%s' not valid for internal reader mode '%s'`	The values specified for Content-Type and internal reader mode are not a supported combination.
4	25	`JCL symbol name '%s' is not valid`	The specified symbol name does not match the syntax rules for a JCL symbol or start with the characters 'SYS'.
4	26	`JCL symbol '%s' value exceeds maximum length`	The value supplied for the specified symbol name exceeds the maximum value length of 255 characters.
4	27	`No value supplied for JCL symbol '%s'`	For a JCL symbol to be defined, it must have a non-null, non-blank value.
4	28	`Maximum number of JCL symbols exceeded`	An attempt to define more than 128 symbols has occurred.
4	29	`User correlator '%s' is not valid`	The specified user correlator (X-IBM-User-Correlator) does not match the syntax rules for a user correlator.
4	30	`Notification URL '%s' exceeds maximum length`	The specified notification URL exceeds the maximum value of 2083 characters.
4	31	`Request header not supported by primary JES subsystem: %s`	The request header identified in the message is not supported by the primary job entry subsystem.
4	32	`Error parsing JSON input`	When submitting a job, Content-Type specified `application/json`, but an exception occurred when attempting to process the input JSON document. Possible causes include: • JCL stream was provided instead of a JSON document • JSON document was malformed • Required “file” property was not provided • Value for the “recall” property was not valid. See the “stack” property of the JSON error report document for a message with additional information.
4	33	`Data set is migrated: %s`	The z/OS data set specified in the JSON document is migrated. No recall is issued. No job was submitted.
4	34	`Recall issued for migrated data set: %s`	The z/OS data set specified in the JSON document is migrated. A recall without waiting has been issued. No job was submitted.

Table 268. Category 6 errors (continued)

rc	reason	message	Description
4	35	`Error recalling data set, RC=%d`	An error occurred when attempting to recall a migrated data set. The return code from the ARCHRCAL service is included in the message. For details on ARCHRCAL and return codes from the DFSMShsm user macros, see *z/OS DFSMShsm Managing Your Own Data*.
4	36	`Incorrect internal reader class: %s. Must be one character in length.`	Internal reader class request header specified a value that is not valid. The class must be one character in length.
4	37	`Incorrect job update version requested: %s.`	The job modify request value for the "version" property or the X-IBM-Job-Modify-Version request header is not valid. The value must be `"1.0"` or `"2.0"`.
8	1	`Unable to query information about submitted job: %s`	The job status for the submitted job could not be obtained within the timeout period (3 seconds).
8	2	`EOF encountered before all requested bytes read (%d / %d)`	Internal read state error. The expected number of bytes were not available to be read before EOF
8	3	`Range start is beyond end of spool file %d for job %s`	During a request to get a range of records for a spool file, the X-IBM-Record-Range header specified a record start value that is beyond the end of the spool file.
8	4	`Cannot advance spool file more than Integer.MAX_VALUE. DD= %s`	During a request to get a range of records for a spool file, the X-IBM-Record-Range header specified a record start value greater than 2**31-1.
8	5	`Error opening input data set: %s`	An error occurred opening the input z/OS data set. See the "stack" property of the JSON error report document for a message with additional information.
8	6	`Error reading submit input data`	An error occurred reading the submit input data. See the "stack" property of the JSON error report document for a message with additional information.
8	7	`Error opening input file: %s`	An error occurred opening the input z/OS UNIX file. See the "stack" property of the JSON error report document for a message with additional information.
8	8	`IAZSYMBL error defining %s`	The JES symbol definition service (IAZSYMBL) failed while trying to define the specified information. In the message, *%s* is one of the following values: • User correlator • Notification URL • One or more JCL symbols. For details about the IAZSYMBL error, see the "stack" property of the JSON error report document.
12	1	`Not authorized to access spool file`	An authorization check failed trying to OPEN the requested spool file.
12	2	`Not authorized to submit job`	Failed an authorization check when attempting to open the internal reader to submit a job.

Table 268. Category 6 errors (continued)

rc	reason	message	Description
12	3	`User not authorized to issue a CIM request`	CIM detected an authentication or authorization failure during the request. For the requested service to be performed, the user must be authorized to use the CIM server and be permitted to the JES2-JES3Jobs CIM provider. The requested service was one of the following: • Hold a job • Release a job • Change the job class • Cancel a job • Delete a job. CIM provides jobs (CFZSEC and CFZRCUST) to help you configure the CIM server, including security authorizations and file system customization. See the chapter on CIM server quick setup and verification in *z/OS Common Information Model User's Guide*.

Category 7 — Unexpected error

Table 269 shows the possible conditions for this error category.

Table 269. Category 7 errors

rc	reason	message	Description
16	1	`Server error occurred`	For details about the exception, see the stack property of the JSON error report document, which is described in "Error report document" on page 500.

Category 8 — SSI extended status error

Table 270 shows the possible conditions for this error category.

Table 270. Category 8 errors

rc	reason	message	Description
n	*m*	varies	The rc and reason (*n*,*m*) are set from the extended status function call of the subsystem interface (SSI Function Code 80) return code and the subsystem options block (SSOB) return code, respectively. The details property of the JSON error report document contains a message with more information. See "Error report document" on page 500.

Category 9 — Common Information Model (CIM) error

Table 271 shows the possible conditions for this error category.

Table 271. Category 9 errors

rc	reason	message	Description
4	2	`Incorrect jobname: "%s"`	Prior to the CIM service call, the job name was found to be null or an empty string.
4	3	`Incorrect jobid: "%s"`	Prior to the CIM service call, the job ID was found to be null or an empty string.

Table 271. Category 9 errors (continued)

rc	reason	message	Description
4	4	`Incorrect JES type`	Prior to the CIM service call, an incorrect JES type (not JES2 or JES3) was detected.
4	5	`Incorrect job class: "%s"`	Prior to the CIM service call, the job class was found to be null or an empty string.
8	—	varies	CIM internal error. An error occurred during setup or invocation of the CIM service.
12	*m*	`Error returned from CIM job {Cancel|Hold|Release| Request Property Change}service`	CIM response error. Reason (*m*) is the reason code returned from CIM. The "details" property of the JSON error report document contains the CIM response text, if any. See "Error report document" on page 500.
16	—	`CIM connection failure`	A connection exception was encountered while processing the requested service. This error can occur during periods of concurrent high usage of these interfaces. Usually, the reason for the failure is a simple connection refused due to overload of the server. The application should try the request again. The number of retry attempts needed depends on how much work is being requested of the server. The requested service was one of the following: • Hold a job • Release a job • Change the job class • Cancel a job • Delete a job.

Category 10 — SSI job modify error

Table 272 shows the possible conditions for this error category.

Table 272. Category 10 errors

rc	reason	message	Description
n	*m*	varies	The rc and reason (*n*,*m*) are set from the job modify function call of the subsystem interface (SSI Function Code 85) return code and the subsystem options block (SSOB) return code, respectively. The "details" property of the JSON error report document contains a message with more information. See "Error report document" on page 500.

z/OSMF information retrieval service

The z/OSMF information retrieval service is an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. This service allows the caller to query the version and other details about the instance of z/OSMF running on a particular system.

z/OSMF information includes the following details:
- SAF realm
- z/OSMF listening port
- z/OSMF version and release
- Installed plug-ins and plug-in build levels
- Indicates the z/OS operating system level.

With this information, a calling program can determine which z/OSMF plug-ins and API functions are available for use on a given system. For information, see "Retrieve z/OSMF information" on page 510.

Required authorizations

None.

Error handling

For errors that occur during the processing of a request, the API returns an appropriate hypertext transfer protocol (HTTP) status code to the calling client. An error is indicated by a 4*nn* code or a 5*nn* code. Some errors might also include a returned JSON object that contains a message that describes the error.

The following HTTP status codes are valid:

HTTP 200 OK
: Success.

HTTP 400 Bad request
: Request contained incorrect parameters.

HTTP 401 Unauthorized
: Submitter of the request did not authenticate to z/OSMF or is not authorized to use the information retrieval service.

HTTP 500 Internal server error
: Programming error.

Error logging

Errors from the information retrieval service are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required.

For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Retrieve z/OSMF information

You can use this operation to retrieve information about z/OSMF on a particular z/OS system.

HTTP method and URI path

```
GET /zosmf/info
```

where **zosmf/info** identifies the z/OSMF information retrieval service.

Standard headers

Use the following standard HTTP header with this request:

```
Content-Type: application/json
```

Custom headers

None.

Request content

None.

Content type used for HTTP response data

The JSON content type (`"Content-Type: application/json"`) is used for response data. The following JSON object is received as output from the request.

```
{
"zosmf_saf_realm":"SAF-profile-prefix",
"zosmf_port":"zosmf-server-port-number",
"zosmf_full_version":"zosmf-release-level",
"plugins":
 [{
 "pluginVersion":"plugin-fmid-build-level",
 "pluginStatus":"plugin-status",
 "pluginDefaultName":"plugin-name"
 },
⋮
  {
 "pluginVersion":"plugin-fmid-build-level",
 "pluginStatus":"plugin-status",
 "pluginDefaultName":"plugin-name"
 }],
"api_version":"api-version",
"zos_version":"zos-release",
"zosmf_version":"zosmf-version",
"zosmf_hostname":"host-system-URL"
}
```

where:

zosmf_saf_realm
: Realm associated with the system on which z/OSMF is installed. Usually, this is the sysplex name.

zosmf_port
: Port number for SSL encrypted traffic for the active instance of z/OSMF on the z/OS system.

zosmf_full_version
Indicates the z/OSMF version, further qualified by a service level.

plugins
Array of zero, one, or more elements that contain information about each of the installed z/OSMF plug-ins. If no plug-ins are installed, this area is empty.

Each element contains the following attributes:

pluginVersion
Indicates the plug-in version (FMID) and build level.

pluginStatus
Indicates the status of the plug-in. The status is reported for IBM-supplied plug-ins only. For an external application, the status is blank.

The following values are valid:

ACTIVE
The plug-in is running.

INSTALLED
The plug-in is installed, but not running.

UNINSTALLED
The plug-in was installed in a previous z/OSMF configuration, but is not installed in the current configuration. This status can result when a plug-in is removed from z/OSMF.

After a plug-in is started, its status remains as ACTIVE, even if the plug-in is later stopped.

pluginDefaultName
Indicates the plug-in name.

api_version
Version of the z/OSMF information retrieval service and the JSON object structure used for this request. The version sequence starts at 1, and is incremented if the service or the JSON structure changes.

zos_version
Indicates the z/OS operating system level. The following values are valid:
04.24.00
Indicates that the z/OS level is V2R1.

zosmf_version
Indicates the z/OSMF level. The following values are valid:
24 Indicates that the z/OSMF level is V2R1.

zosmf_hostname
Indicates the hostname or IP address of the z/OS system on which z/OSMF is installed

Usage considerations

See "Usage considerations for the z/OSMF REST services" on page 2.

Required authorizations

See "Required authorizations" on page 215.

Expected response

On completion, the service returns an HTTP response, which includes a status code indicating whether your request completed. Status code `200` indicates success. A status code of 4*nn* or 5*nn* indicates that an error has occurred. For more details, see "Error handling" on page 509.

The response also includes a JSON object that contains the retrieved data. For details, see "Content type used for HTTP response data" on page 510.

Example request

In the following example, the GET method is used to retrieve information about z/OSMF.

```
GET /zosmf/info HTTP/1.1
Host: host.name.com
```

Figure 273. Sample request to retrieve z/OSMF information

Example response

For a successful request, the HTTP response includes a JSON document containing the requested information.

```
HTTP/1.1 200 OK
Date: Wed, 06 Mar 2013 06:39:28 +0000GMT
Content-Type: text/plain
Connection: close

{
"zosmf_saf_realm":"SAFRealm",
"zosmf_port":"443",
"zosmf_full_version":"24.02",
"plugins":
 [{"pluginVersion":"hsma210.spe2;driver05;2014-02-11T03:21:53",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"Import Manager"},
 {"pluginVersion":"hsma213.spe2;driver05;2014-02-11T10:17:27",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"WorkloadManagement"},
 {"pluginVersion":"HQX7790;driver122;2014-02-18T00:00:00Z",
 "pluginDefaultName":"IBM SDSF"},
 {"pluginVersion":"hsma216;DRIVER04;2014-01-14T19:03:40",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"Capacity Provisioning"},
 {"pluginVersion":"hsma214.spe2;driver05;2014-02-11T08:28:24",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"Software Deployment"},
 {"pluginVersion":"hsma21a;pm93903;2013-08-12T03:52:53",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"ConfigurationAssistant"},
 {"pluginVersion":"hsma215.spe2;driver05;2014-02-11T10:16:30",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"IncidentLog"},
 {"pluginVersion":"hsma211.spe2;driver05;2014-02-11T03:29:52",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"ISPF"},
 {"pluginVersion":"hsma212;DRIVER4B;2014-01-14T12:43:43",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"ResourceMonitoring"},
 {"pluginVersion":"hsma217.spe2;driver05;2014-02-11T12:01:40",
 "pluginStatus":"ACTIVE",
 "pluginDefaultName":"Workflow"}],
"api_version":"1",
"zos_version":"04.24.00",
"zosmf_version":"24",
"zosmf_hostname":"host.name.com"
}
```

z/OSMF system variable services

The z/OSMF system variable services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to create and manage z/OSMF system variables.

Table 273 lists the operations that the system variable services provide.

Table 273. z/OSMF system variable services: operations summary

Operation name	HTTP method and URI path
"Create or update system variables" on page 515	`POST /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>`
"Get system variables" on page 517	`GET /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>`
"Import system variables" on page 519	`POST /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>/actions/import`
"Export system variables" on page 521	`POST /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>/actions/export`
"Delete system variables" on page 523	`DELETE /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>`

Table 274 describes the variables that can be specified in the system variable services URI paths.

Table 274. z/OSMF system variable services: URI path variables

URI path variable	Description
<version>	The version of the system variable services API. The following value is valid: `1.0`.
<sysplex-name>	The name of the sysplex.
<system-name>	The name of the system.

Authorization requirements

Use of the system variable services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

Use of some of the console APIs requires READ access to the following resource profile in the ZMFAPLA class: ZOSMF.VARIABLES.SYSTEM.ADMIN. See the description of the individual APIs for details.

Error logging

Errors from the system variable services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded.

HTTP 204 No content
The request was processed successfully; however, no content was returned. This status is normal for some types of requests, such as creating or updating system variables.

HTTP 400 Bad request
The request was missing required input, had errors in the provided input, included extraneous input, or cannot be otherwise served. Additional information regarding the error is provided in an error response body that includes a reason code with additional information. Do not repeat the request without first correcting it.

HTTP 401 Unauthorized
The request cannot be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both. Or, the client did not authenticate to z/OSMF by using a valid WWW-Authenticate header.

HTTP 404 Not found
The requested resource does not exist.

HTTP 500 Server error
A server error occurred during processing of the request.

Create or update system variables

Use this operation to create or update z/OSMF system variables in the system variable pool.

HTTP method and URI path

```
POST /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF system variables service. The following value is valid: `1.0`.
- *<sysplex-name>* identifies the sysplex.
- *<system-name>* identifies the system.

Description

This operation creates or updates system variables specified in the request body. If the system variable pool does not exist, this operation creates the pool and adds the variables to it. If there is no request body and the pool does not already exist, the operation creates an empty pool. If there is no request body and the pool already exists, no action is taken. If a variable appears in the request body multiple times, the value of the last occurrence is used as the value of the variable.

On successful completion, HTTP status code `204` (No content) is returned, indicating that the system variables were created or updated with the new value.

Authorization requirements

See "Authorization requirements" on page 513.

Request content

The request content is expected to contain an array of JSON objects. See Table 275.

Table 275. Fields in a JSON object for the create or update system variables request

Field name	Type	Required or optional	Description
`name`	String	Required	Descriptive name for the variable
`value`	String	Required	Value for the variable
`description`	String	Optional	Description of the variable

HTTP status codes

On successful completion, HTTP status code `204` (no content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 276. HTTP error response codes for a create or update system variables request

HTTP error status code	Reason Code	Description
`401`		Submitter of the request is not authorized to invoke the task to create the system variables

Table 276. HTTP error response codes for a create or update system variables request (continued)

HTTP error status code	Reason Code	Description
404		Requested *sysplex-name.sysname-name* was not found
404	1	Requested *system-name* was not found

Response content

None.

Example HTTP interaction

In the following example, the POST method is used to create the system variables.

```
POST /zosmf/variables/rest/1.0/variables/systems/sysplex-name.system-name
```

Figure 274. Sample request to create system variables

The request body is as follows:

```
[
  {"name" : "var1","value":"value1","description":"description of the variable"},
  {"name" : "var2","value":"value2","description":"description of the variable"},....
]
```

Figure 275. Sample request from a create system variables request

Get system variables

Use this operation to get all of the z/OSMF system variables.

HTTP method and URI path

```
GET /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF system variables service. The following value is valid: `1.0`.
- *<sysplex-name>* identifies the sysplex.
- *<system-name>* identifies the system.

Query parameters

You can specify the following query parameter on the request to control which system variables are retrieved:

name=*var-name*
Specifies the name of the system variable or variables. The variable *var-name* can be the full name of the system variable or a partial name with the * wildcard character as a suffix or prefix, for example, abc* or *abc. When specifying multiple variable name filters, separate them with &, for example, `?name=abc*&name=*def`

Description

This operation retrieves system variables from the system variable pool and returns them in a list. You can filter the returned list by using a variable name as the query parameter.

On successful completion, HTTP status code `200` is returned, along with a response body, which is described in "Response content" on page 518.

Request content

None.

Authorization requirements

See "Authorization requirements" on page 513.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

If the system variable pool does not exist for the requested system, HTTP status code 200 is returned with an empty array of variables.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 277. HTTP error response codes for a get variables request

HTTP error status code	Reason Code	Description
`401`		The submitter of the request is not authorized to invoke the task to get the system variables

Table 277. HTTP error response codes for a get variables request (continued)

HTTP error status code	Reason Code	Description
404	1	Requested system was not found
404	2	Requested system variable pool was not found

Response content

On successful completion, the service returns a response body, which contains a JSON object with details about the system variables. See Table 278 and Table 279. If no system variables match the filter criteria, HTTP status code 200 (OK) is returned with an empty array.

Table 278. Get system variables request response body

Field name	Type	Description
variables	Array of objects	A list of one or more variables retrieved from system variable pool. See Table 279.

Table 279. Get system variables request: objects

Field name	Type	Description
name	String	Descriptive name for the variable
value	String	Value for the variable
description	String	Description of the variable

Example HTTP interaction

In the following example, the GET method is used to get all of the system variables on a system.

```
GET /zosmf/variables/rest/1.0/variables/systems./<sysplex-name>.<system-name>
```

Figure 276. Sample request to get system variables

```
{variables: [
{"name":"sample1", "value":"20", "Description":"Description":"value of sample1"},
 {...  }
 ]
}
```

Figure 277. Sample response from a get system variables request

Import system variables

Use this operation to import z/OSMF system variables from a file.

HTTP method and URI path

```
POST /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>/actions/import
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF system variables service. The following value is valid: `1.0`.
- *<sysplex-name>* identifies the sysplex.
- *<system-name>* identifies the system.

Description

This operation imports system variables from a file. The file must be accessible by the authenticated user. The file contains variable definitions in comma-separated value (CSV) format, where each row consists of the variable name, value and description. There should be no header row in the file. The variables imported from the file are processed in the same way as variables that are specified with the create system variables API.

On successful completion, HTTP status code `204` (No content) is returned.

Authorization requirements

See "Authorization requirements" on page 513.

Request content

The request content is expected to contain a JSON object. See Table 280.

Table 280. Request content for the import system variables request

Field name	Type	Required or optional	Description
variables-import-file	String	Required	Path to the CSV-formatted file containing the variables to import

HTTP status codes

On successful completion, HTTP status code `204` (No content) is returned.

If the system variable pool does not exist for the requested system, HTTP status code 204 is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 281. HTTP error response codes for a create system variables request

HTTP error status code	Description
`400`	Request body is not syntactically correct
`400`	Specified file was either not found or could not be opened
400	Specified file has an incorrect format

Table 281. HTTP error response codes for a create system variables request (continued)

HTTP error status code	Description
401	Submitter of the request is not authorized to add or delete system variables
404	Requested system was not found
500	A server error occurred during processing of the request.

Response content

None.

Example HTTP interaction

In the following example, the POST method is used to import system variables from a file.

```
POST /zosmf/variables/rest/1.0/systems/TESTPLEX.TESTNODE/actions/import
```

Figure 278. Sample request to import system variables

The request body is as follows:

```
{ "variables-import-file": "/u/testuser/variables.csv" }
```

Figure 279. Sample request body for an import system variables request

Export system variables

Use this operation to export z/OSMF system variables for a specific system to a file.

HTTP method and URI path

```
POST /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>/actions/export
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF system variables service. The following value is valid: `1.0`.
- *<sysplex-name>* identifies the sysplex.
- *<system-name>* identifies the system.

Description

This operation exports system variables, for the system identified in the URI, to a CSV file specified by the request body. It creates the file if it does not exist. Files created by this API can be imported with the import system variables API.

On successful completion, HTTP status code `204` (No content) is returned.

Authorization requirements

See "Authorization requirements" on page 513.

Request content

The request content is expected to contain a JSON object. See Table 282.

Table 282. Request content for the export system variables request

Field name	Type	Required or optional	Description
variables-export-file	String	Required	Path to the file to contain the exported system variables. The file must be accessible to the authenticated user.
overwrite	Boolean	Optional	Indicates whether or not the file should be written to if it already exists. If the value is false and the file exists, the call returns with a status code 400. The value defaults to false if it is not specified.

HTTP status codes

On successful completion, HTTP status code `204` (No content) is returned.

If the system variable pool does not exist for the requested system, HTTP status code 204 is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 283. HTTP error response codes for a create system variables request

HTTP error status code	Description
400	Request body is not syntactically correct
`400`	Path is not accessible for writing

Table 283. HTTP error response codes for a create system variables request (continued)

HTTP error status code	Description
400	File exists, but the request did not indicate that it should be written to
404	Requested system was not found
500	A server error occurred during processing of the request

Response content

None.

Example HTTP interaction

In the following example, the POST method is used to export system variables from a file.

```
POST /zosmf/variables/rest/1.0/systems/TESTPLEX.TESTNODE/actions/export
```

Figure 280. Sample request to export system variables

The request body is as follows:

```
{ "variables-export-file": "/u/testuser/backup-variables.csv", "overwrite":true }
```

Figure 281. Sample request body for an export system variables request

Delete system variables

Use this operation to delete z/OSMF system variables from the system variable pool.

HTTP method and URI path

```
DELETE /zosmf/variables/rest/<version>/systems/<sysplex-name>.<system-name>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF system variables service. The following value is valid: `1.0`.
- *<sysplex-name>* identifies the sysplex.
- *<system-name>* identifies the system.

Description

This operation removes system variables from the system variable pool.

If all variables are removed, the system variable pool is empty. If there is no request body, this operation deletes the system variable pool. If the request body contains an empty array ([ ]), no action is taken. If the request body contains no variables in the array, no action is taken. If the request body contains variables that does not exist in the pool, those variables are ignored.

On successful completion, HTTP status code `204` (No content) is returned.

Authorization requirements

See "Authorization requirements" on page 513.

Request content

The request content is expected to contain an array of strings. Each string represents the name of a system variable to delete.

HTTP status codes

On successful completion, HTTP status code `204` (No content) is returned.

If the system variable pool does not exist for the requested system, HTTP status code 204 is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

Table 284. HTTP error response codes for a create system variables request

HTTP error status code	Description
`400`	Request body is not formatted correctly
`401`	Submitter of the request is not authorized to delete the system variables
`404`	Requested system was not found
`500`	A server error occurred during processing of the request.

Response content

None.

Example HTTP interaction

In the following example, the DELETE method is used to delete system variables from a system variable pool.

```
DELETE /zosmf/variables/rest/1.0/variables/systems/<sysplex-name>.<system-name>
```

Figure 282. Sample request to delete system variables

The request body is as follows:

```
["var1","var2",...]
```

Figure 283. Sample request body for a delete system variables request

z/OSMF workflow services

The z/OSMF workflow services are an application programming interface (API), which is implemented through industry standard Representational State Transfer (REST) services. These services allow the caller to create and manage z/OSMF workflows on a z/OS system.

Table 285 lists the operations that the z/OSMF workflow services provide.

Table 285. z/OSMF workflow services: operations summary

Operation name	HTTP method and URI path
"Create a workflow" on page 528	POST /zosmf/workflow/rest/<*version*>/workflows
"Get the properties of a workflow" on page 533	GET /zosmf/workflow/rest/<*version*>/workflows/<*workflowKey*>
"List the workflows for a system or sysplex" on page 549	GET /zosmf/workflow/rest/<*version*>/workflows
"Start a workflow" on page 552	PUT /zosmf/workflow/rest/<*version*>/workflows/<*workflowKey*>/operations/start
"Cancel a workflow" on page 556	PUT /zosmf/workflow/rest/<*version*>/workflows/<*workflowKey*>/operations/cancel
"Delete a workflow" on page 558	DELETE /zosmf/workflow/rest/<*version*>/workflows/<*workflowKey*>
"Retrieve a workflow definition" on page 560	GET /zosmf/workflow/rest/<*version*>/workflowDefinition
"Archive a workflow instance" on page 571	POST /zosmf/workflow/rest/<*version*>/workflows/<*workflowKey*>/operations/archive
"List the archived workflows for a system" on page 573	GET /zosmf/workflow/rest/<*version*>/archivedworkflows
"Get the properties of an archived workflow" on page 576	GET /zosmf/workflow/rest/<*version*>/archivedworkflows/<*workflowKey*>
"Delete an archived workflow" on page 588	DELETE /zosmf/workflow/rest/<*version*>/archivedworkflows/<*workflowKey*>

Table 286 describes the variables that can be specified in the z/OSMF workflow services URI paths.

Table 286. z/OSMF workflow services: URI path variables

URI path variable	Description
<version>	The version of the z/OSMF workflow services API. The following value is valid: `1.0`.
<workflowKey>	The identifier of a unique instance of a workflow, as returned in the response of the operation that created the workflow.

Authorization requirements

Use of the z/OSMF workflow services API requires the client to be authenticated. For information about client authentication in z/OSMF, see "Authenticating to z/OSMF" on page 1.

In addition, the user's z/OS user ID must have READ access to the following resource profile in the ZMFAPLA class: `<SAF-prefix>.ZOSMF.WORKFLOW.WORKFLOWS`. By default, any user ID with z/OSMF

administrator authority can access the z/OSMF workflow services.

Error response content

For most 4*nn* and 5*nn* HTTP error status codes, additional diagnostic information beyond the HTTP status code is provided in the response body for the request. This information is provided in the form of a JSON object containing the following fields:

Table 287. Error response body elements for the z/OSMF workflow services API

Field name	Type	Description
messageID	String	The message identifier identifying the reason for the error.
messageText	String	The message text that describes the error.

Error logging

Errors from the z/OSMF workflow services are logged in the z/OSMF log. You can use this information to diagnose the problem or provide it to IBM Support, if required. For information about working with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

HTTP status codes

The following HTTP status codes are valid:

HTTP 200 OK
The request succeeded. A response body is provided, which contains the results of the request.

HTTP 201 Created
The request succeeded and resulted in the creation of an object.

HTTP 202 Accepted
The request was successfully validated and is performed asynchronously.

HTTP 204 No content
The request succeeded, but no content is available to be returned.

HTTP 400 Bad request
The request contained incorrect parameters.

HTTP 401 Unauthorized
The request cannot be processed because the client is not authorized. This status is returned if the request contained an incorrect user ID or password, or both. Or, the client did not authenticate to z/OSMF by using a valid WWW-Authenticate header.

HTTP 403 Forbidden
The server received the request, but rejected it.

HTTP 404 Not found
The requested resource does not exist.

HTTP 405 Method not allowed
The requested resource is a valid resource, but an incorrect method was used to submit the request. For example, the request used the POST method when the GET method was expected.

HTTP 408 Request timed out
The client did not produce a request within the allowed time. The request can be submitted again later.

HTTP 409 Request conflict
The request cannot be processed because of conflict in the request, such as an edit conflict between multiple updates.

HTTP 500 Server error
The server encountered an error when it processed the request. For a more specific indication of the error, check the response for a reason code.

HTTP 501 Not implemented
The request specifies an HTTP method that is not recognized by the server.

HTTP 503 Service unavailable
The request cannot be carried out by the server because of a temporary condition. A suggested wait time might be indicated in a Retry-After header, if one is provided in the response. Otherwise, the requestor can treat the response as a `500` response.

HTTP 504 Gateway timeout
The server, which is acting as a gateway or proxy, did not receive a timely response from the server that was specified in the URI path (for example, HTTP, FTP, LDAP) or an auxiliary server (such as DNS). This access is needed to complete the request. For example, the server was not able to start a remote REXX or UNIX shell interface.

Create a workflow

You can use this operation to create a z/OSMF workflow on a z/OS system.

HTTP method and URI path

```
POST /zosmf/workflow/rest/<version>/workflows
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.

Query parameters

None.

Description

This operation creates a workflow, based on the properties that are specified in the request body (a JSON object). For the properties that you can specify, see "Request content" on page 529.

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the creation of a new workflow. The URI path for the workflow is provided in the Location response header and a response body is provided, as described in the "Response content" on page 531.

Workflow access type

By default, general information about the workflow and its steps can be viewed by all users of the Workflows task. If you want to restrict access to a workflow or portions of a workflow, you can do so by specifying an *access type* in the request body for your Create Workflow request. The access type determines which users can view the workflow steps and edit the step notes. The access type is specified on the `accessType` property.

The valid values for the `accessType` property are summarized, as follows:

Public Information about the workflow, including the steps and notes, can be viewed by all users.

Restricted
Information about steps, variables, and notes, is restricted to a subset of users—the workflow owner, step owners, and step assignees. Other users cannot access this information.

Private
Information is restricted to a subset of users, and is further limited among these users. The workflow owner can access information about steps, variables, and notes. Step owners and assignees can retrieve information about the steps for which they are assigned or own, and the associated variables for those steps. Other users cannot access this information.

The `accessType` property is optional. If you omit it from the request body, the workflow is created with public access.

With the exception of workflow notes and step notes, this information is also available to REST API requestors through the Get Workflow Properties service. For the types of information that are restricted by access type, see "Get the properties of a workflow" on page 533.

Request content

The request content is expected to contain a JSON object that describes the workflow to be created. Table 288 lists the fields in the JSON object.

Table 288. Request content for the create workflow request

Field name	Type	Required or optional	Description
`workflowName`	String	Required	Descriptive name for the workflow (up to 100 characters). The name cannot contain the symbols for less-than (<), greater-than (>), or ampersand (&). z/OSMF validates this name to ensure that it is unique across all of the existing workflows.
`workflowDefinitionFile`	String	Required	Location of the workflow definition file. This file is the primary XML file for the workflow definition. Specify this value, as follows: • If the workflow definition file resides in a data set member, specify the fully qualified data set name, including the member name. Ensure that this data set is cataloged. • If the workflow definition file resides in a z/OS UNIX file, specify the fully qualified path name of the file, beginning with the forward slash (/) and including the file name. For example: `/usr/lpp/zosmf/V2R1/samples/workflow_sample_automation.xml`.
`variableInputFile`	String	Optional	Specifies an optional properties file that you can use to pre-specify values for one or more of the variables that are defined in the workflow definition file. Specify this property, as follows: • If the workflow variable input file resides in a data set member, specify the fully qualified data set name, including the member name. Ensure that this data set is cataloged. • If the workflow variable input file resides in a z/OS UNIX file, specify the fully qualified path name of the file, beginning with the forward slash (/) and including the file name. For the format of the contents of the variable input file, see "Providing a workflow variable input file" on page 639.

`variables`	Array of objects	Optional	A list of one or more variables for this workflow. The variables that you specify here take precedence over the variables that are specified in the workflow variable input file.
			
			Specify this property as an array of name-value objects, for example:
			`"variables":`
			`[`
			`{"name":"user_name","value":"IBMUSER"},`
			`{"name":"file_name","value":"textfile.txt"}`
			`]`

Table 288. Request content for the create workflow request (continued)

Field name	Type	Required or optional	Description
`resolveGlobalConflictBy Using`	String	Optional	On creation of the workflow, z/OSMF determines whether any of the variables that are supplied in this request (through the variable input file or variables array) would conflict with existing global variables in the Workflows task. In such cases, this property specifies which version of the variable is used, as follows: • When set to `input`, the global variable conflicts are overridden by the variables in specified input file. The global variable value is updated with the input variable value. Use caution with this setting; your selection will affect any other workflows that refer to the same global variable. • When set to `global`, or omitted, the variable value supplied with the request (through the variable input file or variables array) is ignored and the current global value is used. The default is `global`.
`system`	String	Required	Nickname of the system on which the workflow is to be performed. The nickname is a unique name for the system to differentiate it from existing systems that have the same system and sysplex name. The nickname is 1-40 characters long; the valid characters are alphanumeric characters (A-Z, a-z, and 0-9), hyphens (-), and special characters ($ _ # @). Nicknames are case sensitive; for example, SYSTEM1 and System1 are unique values. If the specified workflow definition file contains an immediate execution step (a template step that runs a program in real time), the workflow must be created on the local system. For information about immediate executions steps, see "Template steps" on page 611.
`owner`	String	Required	User ID of the workflow owner. This user can perform the workflow steps or delegate the steps to other users. Specify a valid user ID, as it is defined to your installation's z/OS security management product, such as RACF. A valid user ID consists of one to eight alphanumeric characters (A-Z, a-z, 0-9, #, $, and @).
`comments`	String	Optional	Specifies any information that you want to associate with the creation of this workflow (up to 500 characters). This information is recorded in the workflow history. Consider including a meaningful comment on the workflow, for example: `This workflow was created through the z/OSMF workflow services REST interface.`
`assignToOwner`	Boolean	Optional	Indicates whether the workflow steps are assigned to the workflow owner when the workflow is created. If you set this property to true, or omit the property, z/OSMF assigns the steps to the user ID that is specified on the property `owner`. If you set this property to false, the workflow steps are left unassigned when the workflow is created. The default is `true`.

Table 288. Request content for the create workflow request (continued)

Field name	Type	Required or optional	Description
accessType	String	Optional	Specifies the access type for the workflow. The access type determines which users can view the workflow steps and edit the step notes, as described in "Workflow access type" on page 528. The following values are valid: • Public • Restricted • Private If you omit this property, the workflow is public, by default.

Authorization requirements

See "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code 201 (Created) is returned and the response body is provided, as described in "Response content."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 289. HTTP error response codes for a create workflow request

HTTP error status code	Description
HTTP 400 Bad request	The request contained incorrect parameters. For example: • The specified workflow name contains errors or is not unique. • Workflow definition file contains errors or does not exist. • Variable input file does not exist. • A validation error occurred for one or more of the following properties: variables, system, owner, or comments. • An incorrect value is specified for the property resolveGlobalConflictByUsing.
HTTP 403 Forbidden	The requestor user ID is not permitted to the workflow definition file or the variable input file.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the service returns the following:

- URI path of the created workflow in the Location response header.
- Response body, which contains a JSON object with details about the workflow. Table 290 on page 532 lists the fields in the JSON object.

Table 290. Response from a create workflow request

Field name	Type	Description
workflowKey	String	Workflow key. A string value, which is generated by z/OSMF to uniquely identify the workflow instance.
workflowDescription	String	Workflow description. This value is obtained from the element workflowDescription in the workflow definition file.
workflowID	String	Workflow ID. A short, arbitrary value that identifies the workflow. This value is obtained from the element workflowID in the workflow definition file.
workflowVersion	String	Version of the workflow definition file. This value is obtained from the element workflowVersion in the workflow definition file.
vendor	String	Name of the vendor that provided the workflow definition file. This value is obtained from the element vendor in the workflow definition file.

Example HTTP interaction

In Figure 284, a request is submitted to create the workflow AutomationExample on the system SY1.

```
POST /zosmf/workflow/rest/1.0/workflows HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Content-Type: application/json
Content-Length: 203
Authorization: Basic em9zbWZhZDp6b3NtZmFk

{
    "workflowName":"AutomationExample",
    "workflowDefinitionFile":"/usr/lpp/zosmf/V2R1/samples/workflow_sample_automation.xml",
    "system":"SY1",
    "owner":"zosmfad",
    "assignToOwner":true,
    "accessType": "Restricted"
}
```

Figure 284. Sample request to create a workflow

A sample response is shown in Figure 285.

```
HTTP/1.1 201 Created
content-length: 210
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
location: /zosmf/workflow/rest/1.0/workflows/d043b5f1-adab-48e7-b7c3-d41cd95fa4b0
date: Wed, 11 Feb 2015 18:29:55 GMT
content-type: application/json; charset=UTF-8

{
    "vendor": "IBM",
    "workflowDescription": "Sample demonstrating the use of automated steps in workflow.",
    "workflowID": "automationSample",
    "workflowKey": "d043b5f1-adab-48e7-b7c3-d41cd95fa4b0",
    "workflowVersion": "1.0"
}
```

Figure 285. Sample response from a create workflow request

Get the properties of a workflow

You can use this operation to retrieve the properties of a z/OSMF workflow.

HTTP method and URI path

```
GET /zosmf/workflow/rest/<version>/workflows/<workflowKey>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the workflow to be queried.

Query parameters

You can specify the following query parameter on this request:

`returnData`
This optional query parameter is used to request information about the workflow steps and variables. Include one or both of the following attributes on the `returnData` parameter:

steps Returns an array of step-info objects; one object for each step in the workflow. Table 294 on page 538 lists the fields in the step-info JSON object.

variables
Returns an array of variable-info objects; one object for each variable that is referenced in the workflow. Table 296 on page 544 lists the fields in the variable-info JSON object.

To specify both attributes, separate the attributes by a comma (','), as follows:

```
returnData=steps,variables
```

Do not enclose the attributes in quotation marks.

The response data is limited by the access type of the workflow. For information, see "Effects of access type on the returned data."

Description

This operation retrieves the properties of a z/OSMF workflow. You can optionally expand the returned information through the specification of query parameters. On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 292 on page 535.

For the format of this information, see the JSON objects that are described in Table 294 on page 538 and Table 296 on page 544.

Authorization requirements

See "Authorization requirements" on page 525.

Effects of access type on the returned data

If you include the optional query parameter `returnData` on the request, the operation can return information about the workflow steps or variables, or both. The amount of data that can be retrieved about the workflow steps or variables can be restricted by the workflow *access type*. This value is specified by the workflow owner at workflow creation time.

Generally, a workflow with a public access type is less restricted in the amount of data that is available to the requestor. A workflow with a restricted or private access type is more secure, and requires the caller user ID to be a workflow owner, step owner, or step assignee to use or access areas in the workflow.

All requestors can retrieve the workflow common properties. This data includes the information that is shown in Table 292 on page 535.

For variables data, the workflow owner can retrieve all of the variable properties for the workflow. The step owner and step assignees can retrieve the variable properties that are associated with steps they own. Other users cannot retrieve this data; requests for details about the variables from these users result in empty arrays being returned.

For steps data, the returned data depends on the type of step data that is requested. The access type allows the workflow creator to distinguish between the following types of steps data.

Step data type	Steps properties	Public access workflow	Restricted access workflow	Private access workflow
Step common properties	• `title` • `name` • `owner` • `stepNumber` • `assignees` • `state` • `skills` • `weight` • `autoEnable` • `hasCalledWorkflow` • `userDefined` • `optional`	This data can be retrieved by all requestors.	This data can be retrieved by: • Workflow owner • Step owner and assignees	This data can be retrieved by: • Workflow owner • Step owner or assignee
Step restricted properties	• `description` • `prereqStep`	This data can be retrieved by all requestors.	This data can be retrieved by: • Workflow owner • Step owner and assignees	The workflow owner can retrieve this data for all steps. The step owner and assignees can retrieve this data for their steps only.

Step data type	Steps properties	Public access workflow	Restricted access workflow	Private access workflow
Step detail properties	• calledInstanceURI • calledWorkflowID • calledWorkflowVersion • calledWorkflowMD5 • calledWorkflowDescription • calledWorkflowDefinitionFile • failedPattern • instructions • instructionsSub • isConditionStep • maxLrecl • output • outputSub • outputVariablesPrefix • procName • regionSize • returnCode • saveAsDataset • saveAsDatasetSub • saveAsUnixFile • saveAsUnixFileSub • scriptParameters • submitAs • successPattern • template • templateSub • timeout • variable-references	The workflow owner can retrieve this data for all steps. The step owner and assignees can retrieve this data for their steps only.	The workflow owner can retrieve this data for all steps. The step owner and assignees can retrieve this data for their steps only.	The workflow owner can retrieve this data for all steps. The step owner and assignees can retrieve this data for their steps only.

HTTP status codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided, as described in Table 292.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 291. HTTP error response codes for a get workflow properties request

HTTP error status code	Description
HTTP 404 Not found	The specified workflow key was not found; the workflow does not exist.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the response body is a JSON object that contains the retrieved data. Table 292 lists the fields in the JSON object.

Table 292. JSON object that is returned to a get workflow properties request

Field name	Type	Description
workflowName	String	Descriptive name for the workflow.
workflowKey	String	Workflow key. A string value, generated by z/OSMF to uniquely identify the workflow instance.
workflowDescription	String	Description of the workflow.

Table 292. JSON object that is returned to a get workflow properties request (continued)

Field name	Type	Description
`workflowID`	String	Workflow ID. A short, arbitrary value that identifies the workflow.
`workflowVersion`	String	Version of the workflow definition file.
`workflowDefinitionFileMD5Value`	String	The 128-bit hash value that is associated with the workflow definition file that was used to create the workflow.
`vendor`	String	Name of the vendor that provided the workflow definition file.
`owner`	String	User ID of the workflow owner.
`system`	String	Full name of the z/OS system on which the workflow is to be performed. This value is in the format *sysplex.sysname*.
`category`	String	Category of the workflow, which is either general or configuration.
`productID`	String	Identifier of the product or component that is being configured through the workflow, such as the product identifier (PID) or function modification identifier (FMID).
`productName`	String	Name of the product or component that is being configured through the workflow.
`productVersion`	String	Version and release of the product or component that is configured through the workflow.
`percentComplete`	Integer	Percentage of the workflow that is completed. z/OSMF calculates this value based on the number of steps in the workflow and the relative weighting value of each step.
`statusName`	String	Indicates the current workflow status, as follows: **in-progress** One or more steps in the workflow are started. **complete** Workflow is complete; all steps are marked complete or skipped. **automation-in-progress** Workflow contains an automated step that is running. **canceled** Workflow is canceled and cannot be resumed. You can, however, view its properties or delete it.
`automationStatus`	Object	An automation-info object that contains details about the most recent start automation request for the workflow. The content of this property depends on the following factors: • If no automation was performed for the workflow, this property is null. • If automation was restarted after being stopped previously, this property indicates the status of the current start automation request. • If automation processing is still in progress, this property indicates which step is being processed. • If automation is stopped and the workflow status is not complete, this property identifies the step most closely related to why automation was stopped. • If automation is stopped and the workflow status is complete, this property indicates that automation is completed. Table 293 on page 537 lists the fields in the automation-info object.

Table 292. JSON object that is returned to a get workflow properties request (continued)

Field name	Type	Description
steps	Array of objects	Array of one or more step-info objects that contain details about each of the steps in the workflow. This property is returned only when the query parameter returnData specifies the attribute steps. The content of this array depends on what the requestor is permitted to see. For more information, see "Description" on page 533. Table 294 on page 538 lists the fields in the step-info object.
variables	Array of objects	Array of one or more variable-info objects that contain details about the variables that are used in the workflow. This property is returned only when the query parameter returnData specifies the attribute variables. The content of this array depends on what the requestor is permitted to see. For more information, see "Description" on page 533. Table 296 on page 544 lists the fields in the variable-info object.

Format of the automation-info object

Table 293 lists the fields in the automation-info JSON object.

Table 293. Get Workflow Properties request: Format of the automation-info object

Field name	Type	Description
startUser	String	User ID of the user who initiated the automation processing.
startedTime	Timestamp	Time that automation processing started. The timestamp data type is used to mean a non-negative Long integer quantity where the value represents a date and time expressed as the number of milliseconds since midnight on January 1, 1970 UTC.
stoppedTime	Timestamp	Time that automation processing stopped. If automation is still in progress, this property is set to null. The timestamp data type is used to mean a non-negative Long integer quantity where the value represents a date and time expressed as the number of milliseconds since midnight on January 1, 1970 UTC.
currentStepName	String	Name of the step that is being processed automatically. Or, the name of the step that caused automation to stop. If automation is stopped and the workflow status is complete, this property is set to null.
currentStepNumber	String	The step number. If automation is stopped and the workflow status is complete, this property is set to null.
currentStepTitle	String	Step title. If automation is stopped and the workflow status is complete, this property is set to null.
messageID	String	Message identifier for the accompanying message. If automation is still in progress, this property is set to null.
messageText	String	Message text that describes the reason that automation is stopped. If automation is still in progress, this property is set to null.

Format of the step-info object

Table 294 lists the fields in the step-info JSON object. Not all of the properties are returned for every step. Some properties are returned or omitted depending on the step type, as noted in the **When returned** column. This information in this column indicates whether valid data is returned for the step, as follows:

All step types
Properties that are returned for all step types.

Calling steps
Properties that are returned for a step that calls another workflow for execution.

Template steps
Properties that are returned for a step that runs a program, such as a batch job, REXX exec, or UNIX shell script.

REST steps
Properties that are returned for a step that issues a REST API request, such as a GET or PUT request.

With regard to returned data, a template step and a REST step are mutually exclusive. The returned information for a template step does not include the properties for a REST step. Likewise, the returned information for a REST step does not include the properties for a template step. To help you identify which steps are REST steps, the step-info object includes the `isRestStep` property, set to true or false.

Table 294. Get Workflow Properties request: Format of the step-info object

Field name	Type	When returned	Description
`name`	String	All step types	Name of the step.
`actualStatusCode`	String	REST steps only	The actual HTTP status code received from the REST API request. To obtain this value, map it to a workflow variable.
`assignees`	String	Calling steps and template steps	Step assignees; one or more user IDs that are assigned to the step. Multiple items are separated by commas.
`autoEnable`	Boolean	All step types	Indicates whether the step can be performed automatically when all prerequisite steps are completed, and no user inputs are required.
`calledInstanceURI`	String	Calling steps only	For a step that calls another workflow for execution, this property contains the URI path of the called workflow instance. You can use this value to retrieve information about the called workflow. This property is null until the step is performed and either a new instance of the called workflow is created or an existing instance is found.
`calledWorkflowID`	String	Calling steps only	This property contains the workflow ID of a workflow definition file; it is used to help locate an existing workflow instance when this step is performed. This property is null when the property `calledWorkflowMD5` is specified.

Table 294. Get Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
`calledWorkflowVersion`	String	Calling steps only	This property contains the workflow version of a workflow definition file; it is used to help locate an existing workflow instance when this step is performed. This property: • Is null when the property `calledWorkflowMD5` is specified • Might be null when the property `calledWorkflowID` is specified.
`calledWorkflowMD5`	String	Calling steps only	This property contains the 128-bit hash value of a workflow definition file; it is used to help locate an existing workflow instance when this step is performed. This property is null when the property `calledWorkflowID` is specified.
`calledWorkflowDescription`	String	Calling steps only	This property contains a description of the workflow to be called, from the point of view of the calling workflow.
`calledWorkflowDefinitionFile`	String	Calling steps only	This property contains the name of the workflow definition file that will be used to create a new workflow if an existing instance is not found when this step is performed. This property might be null.
`description`	String	All step types	Step description.
`expectedStatusCode`	String	REST steps only	The expected HTTP status code from the REST API request. If the `expectedStatusCode` value does not match the `actualStatusCode` value, the workflow step fails. This behavior is similar to what happens when a template step returns a return code that is greater than the allowed maximum return code.
`failedPattern`	Array of strings	Template steps only	Optional regular expression that can be returned for program execution failures. This property might be null.
`hasCalledWorkflow`	Boolean	Calling steps and template steps	Indicates whether this step calls another workflow (true or false). If true, this step is a "calling" step, that is, it calls another workflow for execution. If false, it is a template step. This property is returned only when steps=null, which indicates a leaf step.
`hostname`	String	REST steps only	Indicates the hostname or IP address of the site to which the REST request is directed. For example: `www.ibm.com`.
`httpMethod`	String	REST steps only	Indicates the HTTP method that is used for issuing the REST API request. The possible values are: • GET • PUT • POST • DELETE
`instructions`	String	Template steps only	Detailed instructions on what the user must do to perform the step.

Table 294. Get Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
instructionsSub	Boolean	Template steps only	Indicates whether the step instructions contain variables (true or false).
isConditionStep	Boolean	Calling steps and template steps	Indicates whether this step is a conditional step (true or false).
isRestStep	Boolean	All step types	Indicates whether this step is a REST API step (true or false). When set to true, the following properties contain details about the REST request. Otherwise, these properties are set to null. • actualStatusCode • expectedStatusCode • hostname • hostnameSub • httpMethod • port • portSub • queryParameters • queryParametersSub • requestBody • requestBodySub • schemeName • schemeNameSub • uriPath • uriPathSub The following step properties are not applicable for a REST step and thus, are omitted from the output: • template • templateSub • output • outputSub • saveAsDataset • saveAsDatasetSub • saveAsUnixFile • saveAsUnixFileSub • submitAs • maxLrecl • returnCode
maxLrecl	Integer	Template steps only	For a step that submits a job, this value specifies the maximum record length, in bytes, for the input data for the job. This value is an integer 80 - 1024. The default is 1024.
optional	Boolean	All step types	Indicates whether the step is optional (true or false).
output	String	Template steps only	Indicates the name of the output file produced by the step (a data set or UNIX file). The output file can contain variables and values that are used by subsequent steps.
outputSub	Boolean	Template steps only	Indicates whether the output file name contains variable substitution (true or false).

Table 294. Get Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
outputVariablesPrefix	String	Template steps only	For a step that creates a variable, this property contains a prefix that identifies a string as a variable. This property might be null.
owner	String	Calling steps and template steps	User ID of the step owner.
port	String	REST steps only	Port number that is associated with the REST request.
portSub	Boolean	REST steps only	Indicates whether the port number contains variable substitution (true or false).
prereqStep	Array of strings	All step types	Lists the names of the steps that must be completed before this step can be performed. Up to 499 prerequisite steps can be defined for a step.
procName	String	Template steps only	For a step that runs a program under TSO/E, this property contains the name of the logon procedure that is used to log into the TSO/E address space. If no value was specified for the step, the default is IZUFPROC.
queryParameters	String	REST steps only	For a REST request that includes query parameters, this property contains the query parameters. Otherwise, this property is null.
queryParametersSub	Boolean	REST steps only	This property indicates whether the query parameters contain variable substitution (true or false). Otherwise, this property is null.
regionSize	String	Template steps only	For a step that runs a program under TSO/E, this property contains the region size for the TSO/E address space. If no value was specified for the step, the default is 50000.
requestBody	String	REST steps only	For a REST request that includes a request body, this property contains the request body. Otherwise, this property is null.
requestBodySub	Boolean	REST steps only	This property indicates whether the request body variable substitution (true or false). Otherwise, this property is null.
returnCode	String	Template steps only	For a step that submits a job to run, this property indicates the return code that was returned when the job was submitted.
saveAsDataset	String	Template steps only	Data set name (fully qualified, no quotation marks) that contains the saved JCL.
saveAsDatasetSub	Boolean	Template steps only	Indicates whether the data set name contains variable substitution (true or false).
saveAsUnixFile	String	Template steps only	UNIX file name (absolute name) that contains the saved JCL.
saveAsUnixFileSub	Boolean	Template steps only	Indicates whether the UNIX file name contains variable substitution (true or false).
schemeName	String	REST steps only	The scheme name that is used for the REST request. For example: http.

Table 294. Get Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
schemeNameSub	Boolean	REST steps only	Indicates whether the scheme name contains variable substitution (true or false).
scriptParameters	Array of strings	Template steps only	For a step that runs a program, this property contains the input parameters that can be set by the step owner. This property might be null.
skills	String	Calling steps and template steps	The type of skills that are required to perform the step.
state	String	All step types	State of the step. One of the following status indicators is displayed: • **Unassigned.** The step is in the *Unassigned* state; no users or groups are assigned to the step. • **Assigned.** Users or groups are assigned to the step, but no user accepted ownership of the step. • **Not Ready.** A user accepted ownership of the step, however, a prerequisite step must be completed or a conditional dependency must be satisfied before the step can be performed. • **Ready.** The step is ready to be performed; all prerequisite steps and conditional dependencies are satisfied. • **In Progress.** The step is in progress. For a parent step, a state of *In Progress* means that at least one of the child steps is started, but is not yet complete, overridden, or skipped. For a leaf step, a state of *In Progress* means that the step is started, but is not yet complete, overridden, or skipped. • **Submitted.** The step included a job, which the step owner submitted. • **Complete.** The step was completed. • **Skipped.** The step was bypassed by the step assignee. • **Complete (Override).** The step was marked complete, but the work was performed outside of the Workflows task. • **Failed.** The step included a job that was submitted by the step owner. However, the job failed to complete successfully. • **Conflicts.** The step created an output file for use by a subsequent step. However, values in that file conflict with existing instance or global variables. • **Condition Not Satisfied.** The step is a conditional step, and the condition is not satisfied.
stepNumber	String	All step types	The step number. Steps are numbered to indicate the sequence in which steps are to be performed. For example, the first step in a workflow is 1.

Table 294. Get Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
steps	Array of objects	All step types	For a parent step, this is a nested array of step-info objects. For a leaf step, this property is null. Check this property first before you check the other, non-common step properties. A non-null value here means that the calling step properties are omitted, as are the template step properties and the REST step properties.
submitAs	String	Template steps only	Indicates the type of executable program: JCL job, a REXX exec, or a UNIX shell script, which includes a REXX exec that is written for the UNIX shell environment. The possible values are the following: • "JCL" • "TSO-REXX" • "shell-JCL" • "TSO-REXX-JCL" • "TSO-UNIX-REXX" • "TSO-UNIX-shell"
successPattern	String	Template steps only	Regular expression that is returned for a successful program execution.
template	String	Template steps only	Indicates the template that is used to run a program or batch job (inline or external file).
templateSub	Boolean	Template steps only	Indicates whether template contains variable substitution (true or false). The default is false.
timeout	String	Template steps only	For a step that runs a REXX exec or UNIX shell script, this property contains the maximum amount of time that the program can run before it is ended by a timeout condition.
title	String	All step types	Step title.
uriPath	String	REST steps only	The URI path to use for the REST request.
uriPathSub	Boolean	REST steps only	Indicates whether the URI path contains variable substitution (true or false).
userDefined	Boolean	All step types	Indicates whether the step was added manually to the workflow (true or false). If true, the step was added by the workflow owner, using the **Update Workflow Steps** action in the Workflows table. If false, the step was defined in the workflow definition that was used to create the workflow.
variable-references	Array of objects	Template steps only	An array of variable-reference objects, the format of which is described in Table 295 on page 544.
weight	Integer	Calling steps and template steps	The relative difficulty of the step compared to other steps within this workflow (an integer value 1 - 1000).

Format of the variable-reference object

Table 295 lists the fields in the variable-reference JSON object.

Table 295. Get Workflow Properties request: Format of the variable-reference object

Field name	Type	Description
name	String	Name of the variable.
scope	String	Variable scope, which is either instance or global.

Format of the variable-info object

Table 296 lists the fields in the variable-info JSON object.

Table 296. Get Workflow Properties request: Format of the variable-info object

Field name	Type	Description
name	String	Name of the variable.
scope	String	Variable scope, which is either instance or global.
type	String	Type of variable, which is one of the following values: • boolean • string • number • date • time
value	String	Variable value.
visibility	String	Public or private.

Example HTTP interaction

In the following example, the GET method is used to retrieve information about a workflow. The workflow is uniquely identified by the workflow key, which is represented by the following string value: `26f1fd84-058b-443c-8e06-5ec318ecdb86`. The query parameter `returnData=steps,variables` is included to request more information about the workflow steps and variables.

```
GET /zosmf/workflow/rest/1.0/workflows/26f1fd84-058b-443c-8e06-5ec318ecdb86?returnData=steps,variables
HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 286. Sample request to get workflow properties

An example of the response is shown in the figures that follow.

```
HTTP/1.1 200 OK
content-length: 9155
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 24 Aug 2016 18:30:33 GMT
content-type: application/json; charset=UTF-8

{
    "percentComplete": 0,
    "workflowDefinitionFileMD5Value": "bd1a9b495dfe37ca7837ab7758433bb0",
    "statusName": "in-progress",
    "vendor": "IBM",
    "accountInfo": null,
    "workflowVersion": "1.0",
    "automationStatus": null,
    "steps": [
        {
            "skills": "System Programmer",
            "hasCalledWorkflow": false,
            "weight": "1",
            "instructions": "This step outputs some variables and prints a few words.\n        ",
            "assignees": "zosmft1",
            "state": "Ready",
            "saveAsDataset": null,
            "stepNumber": "1",
            "templateSub": true,
            "submitAs": "TSO-UNIX-shell",
            "saveAsUnixFile": "/u/${instance-st_user}/savedStuff/myScript.sh",
            "userDefined": false,
            "title": "A step that runs a UNIX shell script.",
            "autoEnable": false,
            "variable-references": [
                {
                    "scope": "instance",
                    "name": "st_group"
                },
                {
                    "scope": "instance",
                    "name": "st_user"
                },
                {
                    "scope": "instance",
                    "name": "procNameVariable"
                }
            ],
            "regionSize": 50000,
            "instructionsSub": false,
            "description": "In this step, you submit an inline UNIX shell script for immediate
                            processing \n\t\ton the host system. In this example, the step is
                            expected to complete successfully.\n\t\t",
            "optional": false,
            "name": "TSO-UNIX-shell_Execution",
            "outputSub": false,
            "scriptParameters": "para1",
            "template": "\n#!/bin/sh\necho \"this is a sample to submit shell script to run
                         immediately\"\necho \"the first parameter is
                          :\" $1 \t\necho ${instance-st_user}\necho prefix:st_group = SYS123\
                          necho prefix:st_user = USERS\necho \"This symbol is used to
                          indicate success\"\t\necho \"The program ran successfully !!\"\t\t\n    ",
            "returnCode": null,
            "saveAsDatasetSub": false,
```

Figure 287. Sample response from a get workflow properties request (Part 1 of 4)

```
            "maxLrecl": 1024,
            "outputVariablesPrefix": "prefix:",
            "prereqStep": null,
            "steps": null,
            "isConditionStep": false,
            "isRestStep": false,
            "saveAsUnixFileSub": true,
            "failedPattern": [
                "failed.*"
            ],
            "successPattern": "success.*",
            "procName": "${instance-procNameVariable}",
            "owner": "zosmft1",
            "output": null,
            "timeout": 60
        },
        {
            "skills": "System Programmer",
            "hasCalledWorkflow": false,
            "weight": "1",
            "instructions": "This step outputs some variables and prints a few words.\n        ",
            "assignees": "zosmft1",
            "state": "Ready",
            "saveAsDataset": null,
            "stepNumber": "2",
            "templateSub": true,
            "submitAs": "TSO-UNIX-REXX",
            "saveAsUnixFile": "/u/${instance-st_user}/savedStuff/myScript.sh",
            "userDefined": false,
            "title": "A step that runs a UNIX REXX exec program.",
            "autoEnable": false,
            "variable-references": [
                {
                    "scope": "instance",
                    "name": "st_group"
                },
                {
                    "scope": "instance",
                    "name": "st_user"
                },
                {
                    "scope": "instance",
                    "name": "procNameVariable"
                }
            ],
            "regionSize": 50000,
            "instructionsSub": false,
            "description": "In this step, you submit an inline UNIX REXX exec for immediate
                            processing \n\t\ton the host system. In this example, the step is
                            expected to fail.\n\t\t",
            "optional": false,
            "name": "TSO-UNIX-REXX_Execution",
            "outputSub": false,
            "scriptParameters": "para1",
            "template": "/*  rexx  */\nparse arg arg1\nSAY \"this is a sample to submit UNIX REXX
                         script to run immediately\"\nSAY \"the first parameter is
                         :\" arg1\nSAY ${instance-st_user}\nSAY \"prefix:st_group =\" SYS123\nSAY \
                         "prefix:st_user =\" USERS\nSAY \"This symbol is used to indicate failed\"\n     ",
            "returnCode": null,
            "saveAsDatasetSub": false,
            "maxLrecl": 1024,
            "outputVariablesPrefix": "prefix:",
            "prereqStep": null,
            "steps": null,
            "isConditionStep": false,
            "isRestStep": false,
            "saveAsUnixFileSub": true,
            "failedPattern": [
                "failed.*"
            ],
            "successPattern": "success.*",
            "procName": "${instance-procNameVariable}",
            "owner": "zosmft1",
            "output": null,
            "timeout": 60
        },
```


Figure 288. Sample response from a get workflow properties request (Part 2 of 4)

```
{
            "skills": "System Programmer",
            "hasCalledWorkflow": false,
            "weight": "1",
            "instructions": "This step outputs some variables and prints a few words.\n        ",
            "assignees": "zosmft1",
            "state": "Ready",
            "saveAsDataset": null,
            "stepNumber": "3",
            "templateSub": true,
            "submitAs": "TSO-REXX",
            "saveAsUnixFile": "/u/${instance-st_user}/savedStuff/myScript.sh",
            "userDefined": false,
            "title": "A step that runs a REXX exec program.",
            "autoEnable": false,
            "variable-references": [
                {
                    "scope": "instance",
                    "name": "st_group"
                },
                {
                    "scope": "instance",
                    "name": "st_user"
                },
                {
                    "scope": "instance",
                    "name": "procNameVariable"
                }
            ],
            "regionSize": 50000,
            "instructionsSub": false,
            "description": "In this step, you submit an inline REXX exec for immediate
                            processing \n\t\ton the host system. In this example, the processing
                            is ended by a time-out condition.\n\t\t",
            "optional": false,
            "name": "TSO-TSO-REXX_Execution",
            "outputSub": false,
            "scriptParameters": "para1",
            "template": "/*  rexx  */\nparse arg arg1\nSAY \"this is a sample to submit TSO REXX
                        script to run immediately\"\nSAY \"the first
                        parameter is :\" arg1\nSAY ${instance-st_user}\nSAY \"prefix:st_group =\"
                        SYS123\nSAY \"prefix:st_user =\" USERS\nSAY \"This execution will meets timeout.\"\n  ",
            "returnCode": null,
            "saveAsDatasetSub": false,
            "maxLrecl": 1024,
            "outputVariablesPrefix": "prefix:",
            "prereqStep": null,
            "steps": null,
            "isConditionStep": false,
            "isRestStep": false,
            "saveAsUnixFileSub": true,
            "failedPattern": [
                "failed.*"
            ],
            "successPattern": "success.*",
            "procName": "${instance-procNameVariable}",
            "owner": "zosmft1",
            "output": null,
            "timeout": 60
        }
    ],
    "access": "Public",
    "productVersion": "Version 1",
    "productID": "ABC123",
    "variables": [
        {
            "scope": "instance",
            "visibility": "private",
            "name": "st_user",
            "value": null,
            "type": "string"
        },
```


Figure 289. Sample response from a get workflow properties request (Part 3 of 4)

```
    {
           "scope": "instance",
           "visibility": "private",
           "name": "st_group",
           "value": null,
           "type": "string"
       },

 {
           "scope": "instance",
           "visibility": "private",
           "name": "procNameVariable",
           "value": null,
           "type": "string"
       }
   ],
   "category": "configuration",
   "system": "PLEX1.SY1 (SY1_003)",
   "workflowName": "workflow_sample_program_execution",
   "workflowDescription": "Sample that demonstrates how to run an executable
                           program from a step.\n\t",
   "workflowID": "programExecutionSample",
   "owner": "zosmft1",
   "jobStatement": null,
   "workflowKey": "26f1fd84-058b-443c-8e06-5ec318ecdb86",
   "productName": "Product ABC"
}
```

Figure 290. Sample response from a get workflow properties request (Part 4 of 4)

List the workflows for a system or sysplex

You can use this operation to list the z/OSMF workflows for a system or sysplex.

HTTP method and URI path

```
GET /zosmf/workflow/rest/<version>/workflows
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.

Query parameters

Optionally, your request can include one or more of the following query parameters to filter the results:

`workflowName`
Workflow name. You can specify a regular expression here to match desired workflow names.

`category`
Category of the workflow, which is either general or configuration.

`system`
Nickname of the system on which the workflow is to be performed.

`statusName`
Workflow status, which can be one of the following values:
- in-progress
- complete
- automation-in-progress
- canceled

`owner`
Workflow owner (a valid z/OS user ID).

`vendor`
Name of the vendor that provided the workflow definition file.

Observe the following conventions:
- Query parameters are optional; you can specify one or more query parameters, as needed.
- You use a question mark ('?') to separate the first query parameter from the resource.
- To specify multiple query parameters in combination, use an ampersand (&).

Description

This operation retrieves a list of workflows that match your search criteria. You can filter the returned list of workflows through the specification of query parameters. You can, for example, limit the results to workflows by name, or the workflows for a particular z/OS system.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 297 on page 550.

Authorization requirements

See “Authorization requirements” on page 525.

For the requestor of this service, the response data is limited by the workflow *access type*. This value is selected by the workflow owner at workflow creation time. All workflows with a public access type are

included in the response. To list restricted or private access type workflows, however, the requestor user ID must be a workflow owner, step owner, or step assignee.

For more information, see "Workflow access type" on page 528.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Otherwise, a non-successful HTTP status code is returned and the response body is a standard error response body with the message ID and the associated error message. For the codes that can be returned, see "HTTP status codes" on page 526.

Response content

On successful completion, the response body contains one property, called workflows. This property is an array of workflow-info objects. Table 297 lists the fields in the workflow-info object. If no workflows match the filter criteria, HTTP status code `200` (OK) is returned with an empty array.

Table 297. List workflows request: Format of the workflow-info object

Field name	Type	Description
`workflowName`	String	Descriptive name for the workflow.
`workflowKey`	String	Workflow key. A string value, generated by z/OSMF to uniquely identify the workflow instance.
`workflowDescription`	String	Description of the workflow.
`workflowID`	String	Workflow ID. A short, arbitrary value that identifies the workflow.
`workflowVersion`	String	Version of the workflow definition file.
`workflowDefinitionFileMD5Value`	String	The 128-bit hash value that is associated with the workflow definition file that was used to create the workflow.
`instanceURI`	String	Workflow instance URI path, which you can use to retrieve information about the workflow.
`owner`	String	User ID of the workflow owner.
`vendor`	String	Name of the vendor that provided the workflow definition file.

Example HTTP interaction

In the following example, the GET method is used to list the workflows on a system. Here, the query parameter `workflowName` is included to limit the results to workflows with names that begin with `AutomationExample`.

```
GET /zosmf/workflow/rest/1.0/workflows?workflowName=AutomationExample.* HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 291. Sample request to list workflows

For a successful request, the HTTP response includes a JSON document that contains the requested information. In the following example, one workflow was found to match the request query parameter.

```
HTTP/1.1 200 OK
content-length: 464
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 11 Feb 2015 18:30:34 GMT
content-type: application/json; charset=UTF-8

{
    "workflows": [
        {
            "instanceURI": "/zosmf/workflow/rest/1.0/workflows/d043b5f1-adab-48e7-b7c3-d41cd95fa4b0",
            "owner": "zosmfad",
            "vendor": "IBM",
            "workflowDefinitionFileMD5Value": "a8825b7497793bc620b0edffa8b97cd9",
            "workflowDescription": "Sample demonstrating the use of automated steps in workflow.",
            "workflowID": "automationSample",
            "workflowKey": "d043b5f1-adab-48e7-b7c3-d41cd95fa4b0",
            "workflowName": "AutomationExample|Canceled|1423679433714",
            "workflowVersion": "1.0"
        }
    ]
}
```

Figure 292. Sample response from a list workflows request

Start a workflow

You can use this operation to start a z/OSMF workflow on a z/OS system. The workflow to be started must contain at least one automated step.

HTTP method and URI path

```
PUT /zosmf/workflow/rest/<version>/workflows/<workflowKey>/operations/start
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the workflow to be started.

Query parameters

None.

Description

This operation is used to start an automated workflow, that is, a workflow with at least one step that can be performed automatically. For information about designing an automated workflow, see "Automated steps" on page 627.

On successful completion, HTTP status code `202` (Accepted) is returned. If you specify a notification URL in your request, a JSON object is posted to the URL when automation stops; the format of the object is described in "Content that is posted to the notification URL" on page 554.

This request must be submitted from the user ID of the step owner for the automated step. Otherwise, the request ends with HTTP status code `400` and an error message. Also, if used with a workflow that contains no automated steps, the request ends with HTTP status code `400` and an error message.

Request content

Your request can contain a JSON object that specifies options for starting the workflow. The request content is optional; it can be omitted or be empty. If so, default values are used for any optional properties that can be specified.

Table 298 on page 553 lists the fields in the JSON object.

Table 298. Request content for the start workflow request

Field name	Type	Required or optional	Description
`resolveConflictByUsing`	String	Optional	Indicates how variable conflicts, if any, are to be handled when the Workflows task reads in the output file from a step that runs a REXX exec or UNIX shell script. The following values are valid for this parameter: **outputFileValue** Allow the output file values to override the existing values. This setting applies to instance variables only; global variables are not overridden by variables in the output file. **existingValue** Use the existing variables values instead of the output file values. **leaveConflict** Automation is stopped. The user must resolve the conflict manually. If you omit this property, the default is `outputFileValue`.
`stepName`	String	Optional	The name of the step at which automation is to begin. If you omit this property, automation processing begins with the first step in the workflow.
`performSubsequent`	Boolean	Optional	If the workflow contains any subsequent automated steps, this property indicates whether z/OSMF is to perform the steps. If you set this property to `true`, or omit the property, z/OSMF attempts to perform the automated steps. If you set this property to `false`, z/OSMF attempts to perform the specified step only. The default is `true`.
`notificationUrl`	String	Optional	A notification URL (up to 2000 characters). Depending on the requirements of your application, you might want to receive a notification when the automated step or steps reach an end state (either completion or failure). Suppose, for example, that your application includes a servlet that is to be given control on completion of the automated steps. If so, you can optionally specify a notification URL to be posted when automation ends. Your installation must add this URL to the list of allowable sites. Work with your security administrator to create the appropriate authorization; see "Allowing cross-site access to REST services" on page 3. If specified, this URL receives a JSON object with the result of the automation processing. For the format of the JSON object, see Table 300 on page 554.

Authorization requirements

See "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code 202 (Accepted) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 299. HTTP error response codes for a start workflow request

HTTP error status code	Description
HTTP 400 Bad request	The request contained incorrect parameters. For example: • Incorrect value is specified for the property resolveConflictByUsing. • Specified workflow contains no automated steps. • Request is submitted from a user ID that is not the step owner for the automated step. • Automation processing attempts to begin at a step that is not automated. • Incorrect value is specified for the property stepName.
HTTP 404 Not found	The specified workflow key was not found; the workflow does not exist.
HTTP 409 Request conflict	Request cannot be performed automatically because the specified workflow has one of the following statuses: • Automation-in-progress • Canceled.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Content that is posted to the notification URL

If you specified a notification URL in your request, the URL receives a JSON object when automation stops. Table 300 lists the fields in the JSON object.

Table 300. Structure of the JSON object that is returned to the notification URL

Field name	Type	Description
instanceURI	String	Specifies the URI path of the workflow that was being automated. You can use the URI path to retrieve more details about the workflow.
statusName	String	Indicates the current workflow status after automation has stopped. The following are the possible values that might be returned after a stopped automation: **in-progress** The workflow automation is stopped. The reason is indicated in the messageID and messageText properties. **complete** Workflow is complete. All steps are marked complete or skipped.
startUser	String	User ID of the user who initiated the automation processing.
startedTime	Timestamp	Time that automation processing started.
stoppedTime	Timestamp	Time that automation processing stopped.
currentStepName	String	Name of the step that caused automation to stop. If the workflow status is complete, this property is set to null.

Table 300. Structure of the JSON object that is returned to the notification URL (continued)

Field name	Type	Description
currentStepNumber	String	The step number. If the workflow status is complete, this property is set to null.
currentStepTitle	String	Step title. If the workflow status is complete, this property is set to null.
messageID	String	This property contains the message identifier that helps identify the reason that automation is stopped.
messageText	String	This property contains the message text associated with the message identifier found in messageID.

Example HTTP interaction

In Figure 293, a request is submitted to start a workflow. Here, the workflow is identified by the workflow key, which is the following string value: `d043b5f1-adab-48e7-b7c3-d41cd95fa4b0`. In this example, the request content is empty. As a result, default values are used for any properties that can be specified; see Table 298 on page 553.

```
PUT /zosmf/workflow/rest/1.0/workflows/d043b5f1-adab-48e7-b7c3-d41cd95fa4b0/operations/start HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Content-Type: application/json
Content-Length: 3
Authorization: Basic em9zbWZhZDp6b3NtZmFk

{}
```

Figure 293. Sample request to start a workflow

A sample response is shown in Figure 294.

```
HTTP/1.1 202 Accepted
content-length: 0
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 11 Feb 2015 18:29:55 GMT
content-type: application/json; charset=UTF-8
```

Figure 294. Sample response from a start workflow request

Cancel a workflow

You can use this operation to cancel a z/OSMF workflow on a z/OS system.

HTTP method and URI path

```
PUT /zosmf/workflow/rest/<version>/workflows/<workflowKey>/operations/cancel
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the workflow to be canceled.

Query parameters

None.

Description

This operation is used to cancel a workflow. Canceling a workflow does not undo any actions that were already performed on the system as part of the workflow.

When canceled, the workflow cannot be resumed. You can view the workflow properties through a GET request, as described in "Get the properties of a workflow" on page 533. Also, you can delete a canceled workflow, as described in "Delete a workflow" on page 558.

When canceled, the name of workflow is changed. The name is appended with the text `|Canceled|`, followed by a timestamp (the date and time expressed in milliseconds since midnight on January 1, 1970 UTC). An example is shown in "Example HTTP interaction" on page 557.

This request is failed for a workflow with the status *Automation in Progress*. That is, it is not possible to cancel the workflow while an automated step is running. You must either allow the processing to complete, or you can stop the processing through the Workflows task.

Request content

None.

Authorization requirements

This request is available to the workflow owner only. A cancel request from another user is rejected with HTTP status code `403` (Forbidden) and an appropriate error message in the JSON response object.

For other authorization requirements, see "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned. Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 301. HTTP error response codes for a cancel workflow properties request

HTTP error status code	Description
`HTTP 403 Forbidden`	The request was submitted from a user ID that is not the workflow owner.
`HTTP 404 Not found`	The specified workflow key was not found; the workflow does not exist.
`HTTP 409 Request conflict`	The request cannot be processed because the workflow has the status *Automation in Progress*.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the response body contains one property, workflowName, which specifies the new name of the canceled workflow.

Example HTTP interaction

In the following example, the PUT method is used to cancel an instance of a workflow. Here, the workflow is identified by the workflow key, which is the following string value: `d043b5f1-adab-48e7-b7c3-d41cd95fa4b0`.

```
PUT /zosmf/workflow/rest/1.0/workflows/d043b5f1-adab-48e7-b7c3-d41cd95fa4b0/operations/cancel HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Content-Type: application/json
Content-Length: 0
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 295. Sample request to cancel a workflow

For a successful request, HTTP response code `200` is returned with the canceled workflow name in response body.

```
HTTP/1.1 200 OK
content-length: 59
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 11 Feb 2015 18:30:33 GMT
content-type: application/json; charset=UTF-8

{
    "workflowName": "AutomationExample|Canceled|1423679433714"
}
```

Figure 296. Sample response from a cancel workflow request

Delete a workflow

You can use this operation to remove a z/OSMF workflow from a z/OS system.

HTTP method and URI path

```
DELETE /zosmf/workflow/rest/<version>/workflows/<workflowKey>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the workflow to be deleted.

Query parameters

None.

Description

This operation is used to delete a workflow from z/OSMF, including any notes that accompany the workflow and its steps, and the history log for the workflow. Deleting a workflow does not undo any actions that were performed on the system as part of the workflow. If you delete a workflow, you are responsible for undoing manually any changes on the system that you no longer require. Ensure that all applicable back-out procedures are followed. See your workflow provider for this information.

This request is failed for a workflow with the status *Automation in Progress*. That is, it is not possible to delete the workflow while an automated step is running. You must either allow the processing to complete, or you can stop the processing through the Workflows task.

You cannot delete a *called workflow*, that is, a workflow that is in-progress as a result of being called by another workflow for execution. For design considerations for called workflows, see “Calling steps” on page 621.

Request content

None.

Authorization requirements

This request is available to the workflow owner only. A delete request from another user is rejected with the HTTP status code `403` (Forbidden) and an appropriate error message in the JSON response object.

For other authorization requirements, see “Authorization requirements” on page 525.

HTTP status codes

On successful completion, HTTP status code `204` (No content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 302. HTTP error response codes for a delete workflow request

HTTP error status code	Description
`HTTP 403 Forbidden`	The request was submitted from a user ID that is not the workflow owner.
`HTTP 404 Not found`	The specified workflow key was not found; the workflow does not exist.
`HTTP 409 Request conflict`	The request cannot be processed because the workflow has the status *Automation in Progress*.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

None.

Example HTTP interaction

In the following example, the DELETE method is used to delete a workflow. Here, the workflow is identified by the workflow key, which is the following string value: `d043b5f1-adab-48e7-b7c3-d41cd95fa4b0`.

```
DELETE /zosmf/workflow/rest/1.0/workflows/d043b5f1-adab-48e7-b7c3-d41cd95fa4b0 HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 297. Sample request to delete a workflow

For a successful request, the HTTP response `204` is returned.

```
HTTP/1.1 204 No Content
content-length: 0
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 11 Feb 2015 18:30:34 GMT
content-type: application/json; charset=UTF-8
```

Figure 298. Sample response from a delete workflow request

Retrieve a workflow definition

You can use this operation to retrieve the contents of a z/OSMF workflow definition from a z/OS system.

HTTP method and URI path

```
GET /zosmf/workflow/rest/<version>/workflowDefinition
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.

Query parameters

You can specify the following query parameters on this request:

`definitionFilePath`
Specifies the location of the workflow definition file, which is either a UNIX path name (including the file name) or a fully qualified z/OS data set name. This parameter is required.

`returnData`
Use this optional parameter to request more information about the workflow definition file. Include one or both of the following attributes on the `returnData` parameter:

steps Returns an array of step-definition objects; one object for each step in the workflow. Table 305 on page 562 lists the fields in the step-definition JSON object.

variables
Returns an array of variable-definition objects; one object for each variable that is referenced in the workflow. Table 309 on page 566 lists the fields in the variable-definition JSON object.

To specify both attributes, separate the attributes by a comma (','), as follows:

```
returnData=steps,variables
```

Do not enclose the attributes in quotation marks.

Description

This operation retrieves the content of a z/OSMF workflow definition. You can optionally expand the returned information through the specification of query parameters.

A workflow definition might consist of multiple XML files, including a primary file and possibly other files that are included by the primary file. The workflow definition resides in a z/OS UNIX file system or a z/OS data set.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 304 on page 561. If you include the optional query parameter `returnData` on the request, the operation can return more information about the workflow definition steps or variables, or both. For the format of this information, see the JSON objects that are described in Table 305 on page 562 and Table 309 on page 566.

Authorization requirements

See “Authorization requirements” on page 525.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 304 on page 561.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 303. HTTP error response codes for a retrieve a workflow definition request

HTTP error status code	Description
`HTTP 403 Forbidden`	The requestor user ID is not permitted to the workflow definition file.
`HTTP 404 Not found`	The specified workflow definition file was not found. This resource is specified on the query parameter `definitionFilePath`.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the response body is a JSON object that contains the retrieved data. Table 304 lists the fields in the JSON object.

Table 304. JSON object that is returned to a retrieve a workflow definition request

Field name	Type	Description
`workflowDefaultName`	String	Identifies the default name for the workflow. This value is shown in the in the workflow name field of the Workflows task when the user creates a new workflow instance.
`workflowDescription`	String	Description of the workflow.
`workflowID`	String	Workflow ID. A short, arbitrary value that identifies the workflow.
`workflowVersion`	String	Version of the workflow definition file.
`vendor`	String	Name of the vendor that provided the workflow definition file.
`workflowDefinitionFileMD5Value`	String	A 128-bit hash value that z/OSMF generates to uniquely identify the workflow definition file.
`isCallable`	String	Indicates the callable scope for the workflow, as follows: **system** An instance of this workflow can only be called from a workflow in the same system. **sysplex** An instance of this workflow can be called from a workflow anywhere in the sysplex. This property is null when the workflow cannot be called by another workflow.
`scope`	String	Indicates the singleton scope for the workflow, as follows: **system** A maximum of one instance of this workflow can exist on any one system in the sysplex. **sysplex** A maximum of one instance of this workflow can exist in the sysplex. **none** An existing instance cannot be used. A new instance of this workflow is always created.
`category`	String	Category of the workflow, which is either general or configuration.
`productID`	String	Identifier of the product or component that is being configured through the workflow, such as the product identifier (PID) or function modification identifier (FMID).
`productName`	String	Name of the product or component that is being configured through the workflow.
`productVersion`	String	Version and release of the product or component that is configured through the workflow.

Table 304. JSON object that is returned to a retrieve a workflow definition request (continued)

Field name	Type	Description
steps	Array of objects	Array of one or more step-definition objects that contain details about each of the steps in the workflow definition file. This property is returned only when the query parameter returnData specifies the attribute steps. See the "Format of the step-definition object" section for a list of the fields in the step-definition object.
variables	Array of objects	Array of one or more variable-definition objects that contain details about the variables that are defined or referenced in the workflow definition file. This property is returned only when the query parameter returnData specifies the attribute variables. Table 309 on page 566 lists the fields in the variable-definition object.

Format of the step-definition object

The following tables list the fields included in the step-definition object JSON object:

- Table 305
- Table 306
- Table 307 on page 563

Table 305. Retrieve a workflow definition request: Fields included in every step-definition object

Field name	Type	Description
name	String	Name of the step.
title	String	Step title.
description	String	Step description.
prereqStep	Array of strings	Lists the names of the steps that must be completed before this step can be performed. Up to 499 prerequisite steps can be defined for a step.
optional	Boolean	Indicates whether the step is optional (true or false).
steps	Array of objects	For a parent step, this property is an array of one or more step-definition objects that contain details about each of the steps in the workflow. For a leaf step, this property is null.

Table 306. Retrieve a workflow definition request: Additional fields included in the step-definition object only for a calling step (a step that calls another workflow)

Field name	Type	Description
calledWorkflowDescription	String	For a step that calls another workflow for execution, this property contains the description of the called workflow. This information might include details such as the name and location of the workflow definition file that is used to create the called workflow.
calledWorkflowID	String	For a step that calls another workflow for execution, this property contains the workflow ID for the called workflow.
calledWorkflowMD5	String	For a step that calls another workflow for execution, this property contains the 128-bit hash value that can be used to identify the called workflow.

Table 306. Retrieve a workflow definition request: Additional fields included in the step-definition object only for a calling step (a step that calls another workflow) (continued)

Field name	Type	Description
calledWorkflowDefinitionFile	String	For a step that calls another workflow for execution, this property contains the path name of the workflow definition file for the called workflow.
calledWorkflowVersion	String	For a step that calls another workflow for execution, this property contains the version of the workflow definition file for the called workflow.
callingStepAutoEnable	Boolean	For a step that calls another workflow for execution, this property indicates whether the step is to be performed automatically when all prerequisite steps are completed, and no user inputs are required (true or false).
callingStepWeight	Integer	For a step that calls another workflow for execution, this property indicates the relative difficulty of the step compared to other steps within this workflow (an integer value 1 - 1000).
callingStepSkills	String	For a step that calls another workflow for execution, this property indicates the type of skills that are required to perform the step.

Table 307. Retrieve a workflow definition request: Additional fields included in the step-definition object only for a normal (non-calling) step

Field name	Type	When returned	Description
actualStatusCode	String	REST steps only	The actual HTTP status code received from the REST API request. To obtain this value, map it to a workflow variable.
autoEnable	Boolean	All step types	Indicates whether the step is to be performed automatically when all prerequisite steps are completed, and no user inputs are required (true or false).
expectedStatusCode	String	REST steps only	The expected HTTP status code from the REST API request. If the expectedStatusCode value does not match the actualStatusCode value, the workflow step fails. This behavior is similar to what happens when a template step returns a return code that is greater than the allowed maximum return code.
failedPattern	Array of strings	Template steps only	Optional regular expression that can be returned for program execution failures. This property might be null.
hostname	String	REST steps only	Indicates the hostname or IP address of the site to which the REST request is directed. For example: www.ibm.com.
httpMethod	String	REST steps only	Indicates the HTTP method that is used for issuing the REST API request. The possible values are: • GET • PUT • POST • DELETE
instructions	String	All step types	Detailed instructions on what the user must do to perform the step.

Table 307. Retrieve a workflow definition request: Additional fields included in the step-definition object only for a normal (non-calling) step (continued)

Field name	Type	When returned	Description
isRestStep	Boolean	All step types	Indicates whether this step is a REST API step (true or false). When set to true, the following properties contain details about the REST request. Otherwise, these properties are set to null. • actualStatusCode • expectedStatusCode • hostname • httpMethod • port • propertyMappings • queryParameters • requestBody • schemeName • uriPath The following step properties are not applicable for a REST step, and are therefore omitted from the output: • template • output • saveAsDataset • saveAsUnixFile • submitAs • maxLrecl
maxLrecl	Integer	Template steps only	For a step that submits a job, this value specifies the maximum record length, in bytes, for the input data for the job. This value is an integer 80 - 1024. The default is 1024.
output	String	Template steps only	Indicates the default name of the output file produced by the step (a data set or UNIX file). The output file can contain variables and values that are used by subsequent steps.
outputVariablesPrefix	String	Template steps only	For a step that creates a variable, this property contains a prefix that identifies a string as a variable. This property might be null.
port	String	REST steps only	Port number that is associated with the REST request.
procName	String	Template steps only	For a step that runs a program under TSO/E, this property contains the name of the logon procedure that is used to log into the TSO/E address space. If no value was specified for the step, the default is IZUFPROC.

Table 307. Retrieve a workflow definition request: Additional fields included in the step-definition object only for a normal (non-calling) step (continued)

Field name	Type	When returned	Description

propertyMappings	Array of objects	REST steps only	An array of property mappings, the format of which is:
			`{`
			`"mapFrom": "property",`
			`"mapTo": "variable"`
			`},`
			In the mappings:
			mapFrom Is the property from the REST request. The value of this property is assigned to the specified "mapTo" workflow variable.
			mapTo Is the workflow variable that will be assigned the value from "mapFrom" property.

queryParameters	String	REST steps only	For a REST request that includes query parameters, this property contains the query parameters. Otherwise, this property is null.
regionSize	String	Template steps only	For a step that runs a program under TSO/E, this property contains the region size for the TSO/E address space. If no value was specified for the step, the default is 50000.
requestBody	String	REST steps only	For a REST request that includes a request body, this property contains the request body. Otherwise, this property is null.
saveAsDataset	String	Template steps only	Data set name (fully qualified, no quotation marks) that contains the saved JCL.
saveAsUnixFile	String	Template steps only	UNIX file name (absolute name) that contains the saved JCL.
schemeName	String	REST steps only	The scheme name that is used for the REST request. For example: `http`.
scriptParameters	Array of strings	Template steps only	For a step that runs a program, this property contains the input parameters that can be set by the step owner. This property might be null.
skills	String	All step types	The type of skills that are required to perform the step.
submitAs	String	Template steps only	Indicates the type of executable program: JCL job, a REXX exec, or a UNIX shell script, which includes a REXX exec that is written for the UNIX shell environment. The possible values are the following: • `"JCL"` • `"TSO-REXX"` • `"shell-JCL"` • `"TSO-REXX-JCL"` • `"TSO-UNIX-REXX"` • `"TSO-UNIX-shell"`
successPattern	String	Template steps only	Regular expression that is returned for a successful program execution.
template	String	Template steps only	Indicates the template that is used for a JCL job, a REXX exec program, or a UNIX shell script.

Table 307. Retrieve a workflow definition request: Additional fields included in the step-definition object only for a normal (non-calling) step (continued)

Field name	Type	When returned	Description
timeout	String	Template steps only	For a step that runs a REXX exec or UNIX shell script, this property contains the maximum amount of time that the program can run before it is ended by a timeout condition.
uriPath	String	REST steps only	The URI path to use for the REST request.
variable-specifications	Array of objects	All step types	An array of variable-specification-info objects, the format of which is described in Table 308.
weight	Integer	All step types	The relative difficulty of the step compared to other steps within this workflow (an integer value 1 - 1000).

Format of the variable-specification-info object

lists the fields in the variable-specification-info JSON object.

Table 308. Retrieve a workflow definition request: Format of the variable-specification-info object

Field name	Type	Description
name	String	Name of the variable.
scope	String	Variable scope, which is either instance or global.
required	Boolean	Indicates whether the variable is required (true or false).

Format of the variable-definition object

Table 309 lists the fields in the variable-definition JSON object.

Table 309. Retrieve a workflow definition request: Format of the variable-definition object

Field name	Type	Description
name	String	Name of the variable.
scope	String	Variable scope, which is either instance or global.
abstract	String	A brief description of the variable.
category	String	Name of the logical grouping to which the variable belongs.
choice	Array of strings	The choice value for the variable.
decimalPlaces	Integer	Maximum number of decimal places that can be specified.
default	String	Default value of the variable.
description	String	Description of the variable.
exposeToUser	Boolean	Indicates whether the variable is displayed to the user in the Workflows task.
maxLength	Integer	Maximum length of the variable value.
maxValue	String	Maximum value of the variable.
minLength	Integer	Minimum length of the variable value.
minValue	String	Minimum value of the variable.

Table 309. Retrieve a workflow definition request: Format of the variable-definition object (continued)

Field name	Type	Description
promptAtCreate	Boolean	Indicates whether the user is prompted to specify a value for the variable during the create workflow process.
regularExpression	String	Provides a standard regular expression that constrains the variable value, as an alternative to the available validation types.
requiredAtCreate	Boolean	Indicates whether a value must be specified for the variable during the create workflow process.
type	String	Type of variable.
validationType	String	Specifies the validation type for the variable.
visibility	String	Indicates whether the variable is displayed to the Workflows task user (either public or private).

Example HTTP interaction

In the following example, the GET method is used to retrieve a workflow definition. The location of the workflow definition is specified on the query parameter `definitionFilePath`. The query parameter `returnData=steps,variables` is included to request more information about the workflow steps and variables.

```
GET
/zosmf/workflow/rest/1.0/workflowDefinition
?definitionFilePath=/usr/lpp/zosmf/V2R1/samples/workflow_sample_automation.xml
&returnData=steps,variables HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 299. Sample request to get a workflow definition

An example of the response is shown in the figures that follow.

```
HTTP/1.1 200 OK
content-length: 5822
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 24 Aug 2016 18:30:34 GMT
content-type: application/json; charset=UTF-8

{
    "workflowDefinitionFileMD5Value": "bd1a9b495dfe37ca7837ab7758433bb0",
    "category": "configuration",
    "scope": "none",
    "isCallable": null,
    "workflowID": "programExecutionSample",
    "workflowDescription": "Sample that demonstrates how to run an executable program from a step.\n\t",
    "vendor": "IBM",
    "workflowVersion": "1.0",
    "workflowDefaultName": null,
    "steps": [
        {
            "skills": "System Programmer",
            "template": "\n#!/bin/sh\necho \"this is a sample to submit shell script to run
                        immediately\"\necho \"the first parameter is :\" $1 \t\necho ${instance-st_user}
                        \necho prefix:st_group = SYS123\necho prefix:st_user = USERS\necho \
                         "This symbol is used to indicate success\"\t\necho \"The program ran successfully !!\"\t\t\n  ",
            "weight": 1,
            "maxLrecl": 1024,
            "instructions": "This step outputs some variables and prints a few words.\n        ",
            "variable-specifications": [
                {
                    "scope": "instance",
                    "name": "st_group",
                    "required": true
                },
                {
                    "scope": "instance",
                    "name": "st_user",
                    "required": true
                },
                {
                    "scope": "instance",
                    "name": "procNameVariable",
                    "required": true
                }
            ],
            "outputVariablesPrefix": "prefix:",
            "saveAsDataset": null,
            "isRestStep": false,
            "submitAs": "TSO-UNIX-shell",
            "saveAsUnixFile": "/u/${instance-st_user}/savedStuff/myScript.sh",
            "title": "A step that runs a UNIX shell script.",
            "failedPattern": [
                "failed.*"
            ],
            "autoEnable": "false",
            "regionSize": 50000,
            "successPattern": "success.*",
            "procName": "${instance-procNameVariable}",
            "description": "In this step, you submit an inline UNIX shell script for immediate
                            processing \n\t\ton the host system. In this example, the step is
                            expected to complete successfully.\n\t\t",
            "name": "TSO-UNIX-shell_Execution",
            "optional": false,
            "scriptParameters": "para1",
            "output": null,
            "timeout": 60
        },
```

Figure 300. Sample response from a get workflow definition request (Part 1 of 3)

```
{   "skills": "System Programmer",
            "template": "/*  rexx  */\nparse arg arg1\nSAY \"this is a sample to submit UNIX REXX script
                        to run immediately\"\nSAY \"the first parameter is:
                        \" arg1\nSAY ${instance-st_user}\nSAY \"prefix:st_group =\" SYS123\nSAY
                        \"prefix:st_user =\" USERS\nSAY \"This symbol is used to indicate failed\"\n    ",
            "weight": 1,
            "maxLrecl": 1024,
            "instructions": "This step outputs some variables and prints a few words.\n        ",
            "variable-specifications": [
                {
                    "scope": "instance",
                    "name": "st_group",
                    "required": true
                },
                {
                    "scope": "instance",
                    "name": "st_user",
                    "required": true
                },
                {
                    "scope": "instance",
                    "name": "procNameVariable",
                    "required": true
                }
            ],
            "outputVariablesPrefix": "prefix:",
            "saveAsDataset": null,
            "isRestStep": false,
            "submitAs": "TSO-UNIX-REXX",
            "saveAsUnixFile": "/u/${instance-st_user}/savedStuff/myScript.sh",
            "title": "A step that runs a UNIX REXX exec program.",
            "failedPattern": [
                "failed.*"
            ],
            "autoEnable": "false",
            "regionSize": 50000,
            "successPattern": "success.*",
            "procName": "${instance-procNameVariable}",
            "description": "In this step, you submit an inline UNIX REXX exec for immediate
                           processing \n\t\ton the host system. In this example, the step is
                           expected to fail.\n\t\t",
            "name": "TSO-UNIX-REXX_Execution",
            "optional": false,
            "scriptParameters": "para1",
            "output": null,
            "timeout": 60
        },
        {
            "skills": "System Programmer",
            "template": "/*  rexx  */\nparse arg arg1\nSAY \"this is a sample to submit TSO REXX script
                        to run immediately\"\nSAY \"the first parameter is :\" arg1\nSAY ${instance-st_user}
                        \nSAY \"prefix:st_group =\" SYS123\nSAY \"prefix:st_user =\" USERS\nSAY \
                        "This execution will meet a timeout.\"\n            ",
            "weight": 1,
            "maxLrecl": 1024,
            "instructions": "This step outputs some variables and prints a few words.\n        ",
            "variable-specifications": [
                {
                    "scope": "instance",
                    "name": "st_group",
                    "required": true
                },
                {
                    "scope": "instance",
                    "name": "st_user",
                    "required": true
                },
                {
                    "scope": "instance",
                    "name": "procNameVariable",
                    "required": true
                }
    ],
```

Figure 301. Sample response from a get workflow definition request (Part 2 of 3)

```
   "outputVariablesPrefix": "prefix:",
            "saveAsDataset": null,
            "isRestStep": false,
            "submitAs": "TSO-REXX",
            "saveAsUnixFile": "/u/${instance-st_user}/savedStuff/myScript.sh",
            "title": "A step that runs a REXX exec program.",
            "failedPattern": [
                "failed.*"         ],
            "autoEnable": "false",
            "regionSize": 50000,
            "successPattern": "success.*",
            "procName": "${instance-procNameVariable}",
            "description": "In this step, you submit an inline REXX exec for immediate
                            processing \n\t\ton the host system. In this example, the processing
                            is ended by a time-out condition.\n\t\t",
            "name": "TSO-TSO-REXX_Execution",
            "optional": false,
            "scriptParameters": "para1",
            "output": null,
            "timeout": 60
        }    ],
     "productVersion": "Version 1",
     "productName": "Product ABC",
     "productID": "ABC123",
     "variables": [       {
            "abstract": "User ID for the started task.",
            "scope": "instance",
            "regularExpression": "^[0-9A-Z$#@]{1,8}$",
            "choice": null,
            "visibility": "private",
            "type": "string",
            "decimalPlaces": null,
            "exposeToUser": false,
            "category": "Started",
            "default": "MYSTUSER",
            "promptAtCreate": false,
            "description": "The user ID under whose authority the new started task will run.\n\t",
            "validationType": "USERID",
            "name": "st_user",
            "requiredAtCreate": false
        },
        {
            "abstract": "Group name for the started task.",
            "scope": "instance",
            "regularExpression": "^[A-Z$#@]{1}[0-9A-Z$#@]{0,7}$",
            "choice": null,
            "visibility": "private",
            "type": "string",
            "decimalPlaces": null,
            "exposeToUser": false,
            "category": "Started",
            "default": "SYS1",
            "promptAtCreate": false,
            "description": "The group name under whose authority the started task will run.\n\t",
            "validationType": "GROUP",
            "name": "st_group",
            "requiredAtCreate": false
        },
        {
            "abstract": "Enter a procedure name for running the program.",
            "scope": "instance",
            "choice": null,
            "visibility": "private",
            "type": "string",
            "decimalPlaces": null,
            "exposeToUser": false,
            "category": "TSO procName",
            "default": null,
            "maxLength": 1000000,
            "promptAtCreate": false,
            "description": "This value is used to specify a procedure name (proc name) for the
                            TSO/E address space \n\tthat is used to run the program.\n\t",
            "name": "procNameVariable",
            "requiredAtCreate": false        }
    ]
}
```

Figure 302. Sample response from a get workflow definition request (Part 3 of 3)

Archive a workflow instance

You can use this operation to archive a z/OSMF workflow instance on a z/OS system.

HTTP method and URI path

```
POST /zosmf/workflow/rest/<version>/workflows/<workflowKey>/operations/archive
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the workflow to be archived.

Query parameters

None.

Description

This operation archives a workflow instance, which is identified by the workflow key that is specified in the request URI.

You can archive any workflows that are completed or that you no longer need. Doing so removes the workflow from the Workflows table in the Workflows task and places it in an archive for your reference. An archived workflow is no longer active, but its information can be viewed by you at any time. When you no longer want to retain an archived workflow, you can delete it permanently from z/OSMF.

After you archive a workflow, you can only list or delete it, or retrieve the workflow properties. You cannot undo this action.

To be archived, a workflow must be in one of the following states:

- In-progress
- Complete
- Canceled.

It is not possible to archive a workflow while it is involved in a workflow activity. Specifically, you cannot archive a workflow when it:

- Is locked for an update operation
- Contains an automated step that is running
- Is waiting for a called workflow to complete
- Is called for processing by another workflow.

To do so, you must allow the processing to complete first.

On successful completion, HTTP status code `201` (Created) is returned, indicating that the request resulted in the archival of the workflow. The URI path for the workflow is provided in the Location response header and a response body is provided, as described in the "Response content" on page 572.

Request content

None.

Authorization requirements

See "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code 201 (Created) is returned and the response body is provided, as described in "Response content."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 310. HTTP error response codes for an archive workflow request

HTTP error status code	Description
HTTP 400 Bad request	The request contained an incorrect parameter, such as an incorrect workflow key.
HTTP 403 Forbidden	The requestor user ID is not permitted to archive the workflow.
HTTP 404 Not found	The specified workflow key was not found; the workflow does not exist.
HTTP 409 Request conflict	Request cannot be performed because the specified workflow has a status that makes the workflow ineligible to be archived.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the service returns the response body, which contains a JSON object with the workflow key. Table 311 describes the contents of the response body.

Table 311. Response from an archive workflow request

Field name	Type	Description
workflowKey	String	Workflow key. A string value, which is generated by z/OSMF to uniquely identify the archived workflow instance.

Example HTTP interaction

In Figure 303, a request is submitted to archive the workflow that is identified by the workflow key 2535b19e-a8c3-4a52-9d77-e30bb920f912.

```
POST /zosmf/workflow/rest/1.0/workflows/2535b19e-a8c3-4a52-9d77-e30bb920f912/operations/archive
Host: zosmf1.yourco.com
Connection: close
```

Figure 303. Sample request to archive a workflow

A sample response is shown in Figure 304.

```
HTTP/1.1 201 Created
{
"workflowKey": "2535b19e-a8c3-4a52-9d77-e30bb920f912"
}
```

Figure 304. Sample response from an archive workflow request

List the archived workflows for a system

You can use this operation to list the archived z/OSMF workflows for a system or sysplex.

HTTP method and URI path

```
GET /zosmf/workflow/rest/<version>/archivedworkflows
```

In this request, the URI path variable *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.

Query parameters

Optionally, your request can include one or more of the following query parameters to filter the results:

`Orderby`

To sort the returned instances by time, specify either of the following values:

"desc":
From the newest to the oldest

"asc": From the oldest to the newest

`View`

An string type to select the list instances by view:

"user":
Return the archived workflow instances that are owned by the user, up to a maximum of 200 workflow instances. This value is the default.

"domain":
For archived provisioning workflows, return the instances that the user is authorized to view. The user must be a domain owner. The results are grouped by domain, with up to 200 instances per domain.

Observe the following conventions:

- Query parameters are optional; you can specify one or more query parameters, as needed.
- You use a question mark ('?') to separate the first query parameter from the resource.
- To specify multiple query parameters in combination, use an ampersand (&).

Description

This operation retrieves a list of archived workflows that you are authorized to view.

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 313 on page 574.

Authorization requirements

See "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 312. HTTP error response codes for a get archived workflow properties request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained an error, such as an incorrect query parameter.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the response body contains one property, which is called archived workflows. This property is an array of workflow-info objects. Table 313 lists the fields in the workflow-info object. If no workflows match the filter criteria, HTTP status code `200` (OK) is returned with an empty array.

Table 313. List archived workflows request: Format of the workflow-info object

Field name	Type	Description
`workflowName`	String	Descriptive name for the workflow.
`workflowKey`	String	Workflow key. A string value, generated by z/OSMF to uniquely identify the workflow instance.
`archivedInstanceURI`	String	Workflow instance URI path, which you can use to retrieve information about the archived workflow.

Example HTTP interaction

In the following example, the GET method is used to list the archived workflows on a system. Here, the query parameter `?orderBy=desc` is included to order the results in descending order.

```
GET /zosmf/workflow/rest/1.0/archivedworkflows?orderBy=desc HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 305. Sample request to list archived workflows

For a successful request, the HTTP response includes a JSON document that contains the requested information. In the following example, three archived workflows were found for the requestor user ID.

```
HTTP/1.1 200 OK
content-length: 464
content-language: en-US
x-powered-by: Servlet/3.0
server: WebSphere Application Server
connection: Close
date: Wed, 11 Feb 2015 18:30:34 GMT
content-type: application/json; charset=UTF-8

HTTP/1.1 200 OK
{
 "archivedWorkflows": [
    {
      "workflowName": "Sample demonstrating variable substitution and the use of a wizard. - Workflow_5",
      "workflowKey": "2535b19e-a8c3-4a52-9d77-e30bb920f912",
      "archivedInstanceURI": "\/zosmf\/workflow\/rest\/1.0\/archivedworkflows\
                    /2535b19e-a8c3-4a52-9d77-e30bb920f912"
       },
    {
       "workflowName": "Sample demonstrating variable substitution and the use of a wizard. - Workflow_0",
       "workflowKey": "8f0f572a-0eb5-493e-91b2-3d549374e07d",
       "archivedInstanceURI": "\/zosmf\/workflow\/rest\/1.0\/archivedworkflows\
                   /8f0f572a-0eb5-493e-91b2-3d549374e07d"
       },
   {
       "workflowName": "Sample demonstrating variable substitution and the use of a wizard. - Workflow_5",
       "workflowKey": "1aead54d-3507-4473-9cda-e4fd25eb21b8",
       "archivedInstanceURI": "\/zosmf\/workflow\/rest\/1.0\/archivedworkflows\
                   /1aead54d-3507-4473-9cda-e4fd25eb21b8"
       }
   ]
}
```

Figure 306. Sample response from a list archived workflows request

Get the properties of an archived workflow

You can use this operation to retrieve the properties of an archived z/OSMF workflow.

HTTP method and URI path

```
GET /zosmf/workflow/rest/<version>/archivedworkflows/<workflowKey>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the archived workflow to be queried.

Query parameters

You can specify the following query parameter on this request:

returnData
This optional query parameter is used to request information about the workflow steps and variables. Include one or both of the following attributes on the `returnData` parameter:

steps Returns an array of step-info objects; one object for each step in the workflow. Table 317 on page 580 lists the fields in the step-info JSON object.

variables
Returns an array of variable-info objects; one object for each variable that is referenced in the workflow. Table 318 on page 585 lists the fields in the variable-info JSON object.

To specify both attributes, separate the attributes by a comma (','), as follows:

```
returnData=steps,variables
```

Do not enclose the attributes in quotation marks.

Description

This operation retrieves the properties of an archived z/OSMF workflow. You can optionally expand the returned information through the specification of query parameters. On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 315 on page 577.

For the format of this information, see the JSON objects that are described in Table 317 on page 580 and Table 318 on page 585.

Authorization requirements

See "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code `200` (OK) is returned and the response body is provided, as described in Table 315 on page 577.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 314. HTTP error response codes for a get archived workflow properties request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained an incorrect parameter, such as an incorrect workflow key.
`HTTP 403 Forbidden`	The requestor user ID is not permitted to retrieve the workflow properties.
`HTTP 404 Not found`	The specified workflow key was not found; the workflow does not exist.

Additional standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

On successful completion, the response body is a JSON object that contains the retrieved data. Table 315 lists the fields in the JSON object.

Table 315. JSON object that is returned to a get archived workflow properties request

Field name	Type	Description
`workflowName`	String	Descriptive name for the workflow.
`workflowKey`	String	Workflow key. A string value, generated by z/OSMF to uniquely identify the workflow instance.
`workflowDescription`	String	Description of the workflow.
`workflowID`	String	Workflow ID. A short, arbitrary value that identifies the workflow.
`workflowVersion`	String	Version of the workflow definition file.
`workflowDefinitionFileMD5Value`	String	The 128-bit hash value that is associated with the workflow definition file that was used to create the workflow.
`vendor`	String	Name of the vendor that provided the workflow definition file.
`owner`	String	User ID of the workflow owner.
`system`	String	Full name of the z/OS system on which the workflow is to be performed. This value is in the format *sysplex.sysname*.
`category`	String	Category of the workflow, which is either general or configuration.
`productID`	String	Identifier of the product or component that is being configured through the workflow, such as the product identifier (PID) or function modification identifier (FMID).
`productName`	String	Name of the product or component that is being configured through the workflow.
`productVersion`	String	Version and release of the product or component that is configured through the workflow.
`percentComplete`	Integer	Percentage of the workflow that is completed. z/OSMF calculates this value based on the number of steps in the workflow and the relative weighting value of each step.
`statusName`	String	Indicates the current workflow status, which is *archived*.

Table 315. JSON object that is returned to a get archived workflow properties request (continued)

Field name	Type	Description
automationStatus	Object	An automation-info object that contains details about the most recent start automation request for the workflow. The content of this property depends on the following factors: • If no automation was performed for the workflow, this property is null. • If automation was restarted after being stopped previously, this property indicates the status of the current start automation request. • If automation processing is still in progress, this property indicates which step is being processed. • If automation is stopped and the workflow status is not complete, this property identifies the step most closely related to why automation was stopped. • If automation is stopped and the workflow status is complete, this property indicates that automation is completed. Table 316 on page 579 lists the fields in the automation-info object.
access	String	Specifies the access type for the workflow. The access type determines which users can view the workflow steps and edit the step notes, as described in "Workflow access type" on page 528. The following values are valid: • Public • Restricted • Private
archivedTime	String	Date and time on the system when the workflow was archived. The date and time is in Greenwich Mean Time (GMT).
accountInfo	String	May be null, or the account information in the JCL JOB statement.
jobStatement	String	May be null, or a list of JCL cards, each up to 72 characters long.
steps	Array of objects	Array of one or more step-info objects that contain details about each of the steps in the workflow. This property is returned only when the query parameter returnData specifies the attribute steps. The content of this array depends on what the requestor is permitted to see. For more information, see "Description" on page 576. Table 317 on page 580 lists the fields in the step-info object.
variables	Array of objects	Array of one or more variable-info objects that contain details about the variables that are used in the workflow. This property is returned only when the query parameter returnData specifies the attribute variables. The content of this array depends on what the requestor is permitted to see. For more information, see "Description" on page 576. Table 319 on page 586 lists the fields in the variable-info object.

Format of the automation-info object

Table 316 lists the fields in the automation-info JSON object.

Table 316. Get Archived Workflow Properties request: Format of the automation-info object

Field name	Type	Description
`startUser`	String	User ID of the user who initiated the automation processing.
`startedTime`	Timestamp	Time that automation processing started. The timestamp data type is used to mean a non-negative Long integer quantity where the value represents a date and time expressed as the number of milliseconds since midnight on January 1, 1970 UTC.
`stoppedTime`	Timestamp	Time that automation processing stopped. If automation is still in progress, this property is set to null. The timestamp data type is used to mean a non-negative Long integer quantity where the value represents a date and time expressed as the number of milliseconds since midnight on January 1, 1970 UTC.
`currentStepName`	String	Name of the step that is being processed automatically. Or, the name of the step that caused automation to stop. If automation is stopped and the workflow status is complete, this property is set to null.
`currentStepNumber`	String	The step number. If automation is stopped and the workflow status is complete, this property is set to null.
`currentStepTitle`	String	Step title. If automation is stopped and the workflow status is complete, this property is set to null.
`messageID`	String	Message identifier for the accompanying message. If automation is still in progress, this property is set to null.
`messageText`	String	Message text that describes the reason that automation is stopped. If automation is still in progress, this property is set to null.

Format of the step-info object

Table 317 on page 580 lists the fields in the step-info JSON object. Not all of the properties are returned for every step. Some properties are returned or omitted depending on the step type, as noted in the **When returned** column. This information in this column indicates whether valid data is returned for the step, as follows:

All step types
: Properties that are returned for all step types.

Calling steps
: Properties that are returned for a step that calls another workflow for execution.

Template steps
: Properties that are returned for a step that runs a program, such as a batch job, REXX exec, or UNIX shell script.

REST steps
: Properties that are returned for a step that issues a REST API request, such as a GET or PUT request.

With regard to returned data, a template step and a REST step are mutually exclusive. The returned information for a template step does not include the properties for a REST step. Likewise, the returned

information for a REST step does not include the properties for a template step. To help you identify which steps are REST steps, the step-info object includes the `isRestStep` property, set to true or false.

Table 317. Get Archived Workflow Properties request: Format of the step-info object

Field name	Type	When returned	Description
`name`	String	All step types	Name of the step.
`actualStatusCode`	String	REST steps only	The actual HTTP status code received from the REST API request. To obtain this value, map it to a workflow variable.
`assignees`	String	Calling steps and template steps	Step assignees; one or more user IDs that are assigned to the step. Multiple items are separated by commas.
`autoEnable`	Boolean	All step types	Indicates whether the step can be performed automatically when all prerequisite steps are completed, and no user inputs are required.
`calledInstanceURI`	String	Calling steps only	For a step that calls another workflow for execution, this property contains the URI path of the called workflow instance. You can use this value to retrieve information about the called workflow. This property is null until the step is performed and either a new instance of the called workflow is created or an existing instance is found.
`calledWorkflowID`	String	Calling steps only	This property contains the workflow ID of a workflow definition file; it is used to help locate an existing workflow instance when this step is performed. This property is null when the property `calledWorkflowMD5` is specified.
`calledWorkflowVersion`	String	Calling steps only	This property contains the workflow version of a workflow definition file; it is used to help locate an existing workflow instance when this step is performed. This property: • Is null when the property `calledWorkflowMD5` is specified • Might be null when the property `calledWorkflowID` is specified.
`calledWorkflowMD5`	String	Calling steps only	This property contains the 128-bit hash value of a workflow definition file; it is used to help locate an existing workflow instance when this step is performed. This property is null when the property `calledWorkflowID` is specified.
`calledWorkflowDescription`	String	Calling steps only	This property contains a description of the workflow to be called, from the point of view of the calling workflow.
`calledWorkflowDefinitionFile`	String	Calling steps only	This property contains the name of the workflow definition file that will be used to create a new workflow if an existing instance is not found when this step is performed. This property might be null.
`description`	String	All step types	Step description.

Table 317. Get Archived Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
expectedStatusCode	String	REST steps only	The expected HTTP status code from the REST API request. If the expectedStatusCode value does not match the actualStatusCode value, the workflow step fails. This behavior is similar to what happens when a template step returns a return code that is greater than the allowed maximum return code.
failedPattern	Array of strings	Template steps only	Optional regular expression that can be returned for program execution failures. This property might be null.
hasCalledWorkflow	Boolean	Calling steps and template steps	Indicates whether this step calls another workflow (true or false). If true, this step is a "calling" step, that is, it calls another workflow for execution. If false, it is a template step. This property is returned only when steps=null, which indicates a leaf step.
hostname	String	REST steps only	Indicates the hostname or IP address of the site to which the REST request is directed. For example: www.ibm.com.
httpMethod	String	REST steps only	Indicates the HTTP method that is used for issuing the REST API request. The possible values are: • GET • PUT • POST • DELETE
instructions	String	Template steps only	Detailed instructions on what the user must do to perform the step.
instructionsSub	Boolean	Template steps only	Indicates whether the step instructions contain variables (true or false).
isConditionStep	Boolean	Calling steps and template steps	Indicates whether this step is a conditional step (true or false).

Table 317. Get Archived Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
isRestStep	Boolean	All step types	Indicates whether this step is a REST API step (true or false). When set to true, the following properties contain details about the REST request. Otherwise, these properties are set to null. • actualStatusCode • expectedStatusCode • hostname • hostnameSub • httpMethod • port • portSub • queryParameters • queryParametersSub • requestBody • requestBodySub • schemeName • schemeNameSub • uriPath • uriPathSub The following step properties are not applicable for a REST step and thus, are omitted from the output: • template • templateSub • output • outputSub • saveAsDataset • saveAsDatasetSub • saveAsUnixFile • saveAsUnixFileSub • submitAs • maxLrecl • returnCode
maxLrecl	Integer	Template steps only	For a step that submits a job, this value specifies the maximum record length, in bytes, for the input data for the job. This value is an integer 80 - 1024. The default is 1024.
optional	Boolean	All step types	Indicates whether the step is optional (true or false).
output	String	Template steps only	Indicates the name of the output file produced by the step (a data set or UNIX file). The output file can contain variables and values that are used by subsequent steps.
outputSub	Boolean	Template steps only	Indicates whether the output file name contains variable substitution (true or false).
outputVariablesPrefix	String	Template steps only	For a step that creates a variable, this property contains a prefix that identifies a string as a variable. This property might be null.
owner	String	Calling steps and template steps	User ID of the step owner.

Table 317. Get Archived Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
port	String	REST steps only	Port number that is associated with the REST request.
portSub	Boolean	REST steps only	Indicates whether the port number contains variable substitution (true or false).
prereqStep	Array of strings	All step types	Lists the names of the steps that must be completed before this step can be performed. Up to 499 prerequisite steps can be defined for a step.
procName	String	Template steps only	For a step that runs a program under TSO/E, this property contains the name of the logon procedure that is used to log into the TSO/E address space. If no value was specified for the step, the default is IZUFPROC.
queryParameters	String	REST steps only	For a REST request that includes query parameters, this property contains the query parameters. Otherwise, this property is null.
queryParametersSub	Boolean	REST steps only	This property indicates whether the query parameters contain variable substitution (true or false). Otherwise, this property is null.
regionSize	String	Template steps only	For a step that runs a program under TSO/E, this property contains the region size for the TSO/E address space. If no value was specified for the step, the default is 50000.
requestBody	String	REST steps only	For a REST request that includes a request body, this property contains the request body. Otherwise, this property is null.
requestBodySub	Boolean	REST steps only	This property indicates whether the request body variable substitution (true or false). Otherwise, this property is null.
returnCode	String	Template steps only	For a step that submits a job to run, this property indicates the return code that was returned when the job was submitted.
saveAsDataset	String	Template steps only	Data set name (fully qualified, no quotation marks) that contains the saved JCL.
saveAsDatasetSub	Boolean	Template steps only	Indicates whether the data set name contains variable substitution (true or false).
saveAsUnixFile	String	Template steps only	UNIX file name (absolute name) that contains the saved JCL.
saveAsUnixFileSub	Boolean	Template steps only	Indicates whether the UNIX file name contains variable substitution (true or false).
schemeName	String	REST steps only	The scheme name that is used for the REST request. For example: http.
schemeNameSub	Boolean	REST steps only	Indicates whether the scheme name contains variable substitution (true or false).
scriptParameters	Array of strings	Template steps only	For a step that runs a program, this property contains the input parameters that can be set by the step owner. This property might be null.

Table 317. Get Archived Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
skills	String	Calling steps and template steps	The type of skills that are required to perform the step.
state	String	All step types	State of the step. One of the following status indicators is displayed: • **Unassigned.** The step is in the *Unassigned* state; no users or groups are assigned to the step. • **Assigned.** Users or groups are assigned to the step, but no user accepted ownership of the step. • **Not Ready.** A user accepted ownership of the step, however, a prerequisite step must be completed or a conditional dependency must be satisfied before the step can be performed. • **Ready.** The step is ready to be performed; all prerequisite steps and conditional dependencies are satisfied. • **In Progress.** The step is in progress. For a parent step, a state of *In Progress* means that at least one of the child steps is started, but is not yet complete, overridden, or skipped. For a leaf step, a state of *In Progress* means that the step is started, but is not yet complete, overridden, or skipped. • **Submitted.** The step included a job, which the step owner submitted. • **Complete.** The step was completed. • **Skipped.** The step was bypassed by the step assignee. • **Complete (Override).** The step was marked complete, but the work was performed outside of the Workflows task. • **Failed.** The step included a job that was submitted by the step owner. However, the job failed to complete successfully. • **Conflicts.** The step created an output file for use by a subsequent step. However, values in that file conflict with existing instance or global variables. • **Condition Not Satisfied.** The step is a conditional step, and the condition is not satisfied.
stepNumber	String	All step types	The step number. Steps are numbered to indicate the sequence in which steps are to be performed. For example, the first step in a workflow is 1.
steps	Array of objects	All step types	For a parent step, this is a nested array of step-info objects. For a leaf step, this property is null. Check this property first before you check the other, non-common step properties. A non-null value here means that the calling step properties are omitted, as are the template step properties and the REST step properties.

Table 317. Get Archived Workflow Properties request: Format of the step-info object (continued)

Field name	Type	When returned	Description
submitAs	String	Template steps only	Indicates the type of executable program: JCL job, a REXX exec, or a UNIX shell script, which includes a REXX exec that is written for the UNIX shell environment. The possible values are the following: • "JCL" • "TSO-REXX" • "shell-JCL" • "TSO-REXX-JCL" • "TSO-UNIX-REXX" • "TSO-UNIX-shell"
successPattern	String	Template steps only	Regular expression that is returned for a successful program execution.
template	String	Template steps only	Indicates the template that is used to run a program or batch job (inline or external file).
templateSub	Boolean	Template steps only	Indicates whether template contains variable substitution (true or false). The default is false.
timeout	String	Template steps only	For a step that runs a REXX exec or UNIX shell script, this property contains the maximum amount of time that the program can run before it is ended by a timeout condition.
title	String	All step types	Step title.
uriPath	String	REST steps only	The URI path to use for the REST request.
uriPathSub	Boolean	REST steps only	Indicates whether the URI path contains variable substitution (true or false).
userDefined	Boolean	All step types	Indicates whether the step was added manually to the workflow (true or false). If true, the step was added by the workflow owner, using the **Update Workflow Steps** action in the Workflows table. If false, the step was defined in the workflow definition that was used to create the workflow.
variable-references	Array of objects	Template steps only	An array of variable-reference objects, the format of which is described in Table 318.
weight	Integer	Calling steps and template steps	The relative difficulty of the step compared to other steps within this workflow (an integer value 1 - 1000).

Format of the variable-reference object

Table 318 lists the fields in the variable-reference JSON object.

Table 318. Get Archived Workflow Properties request: Format of the variable-reference object

Field name	Type	Description
name	String	Name of the variable.
scope	String	Variable scope, which is either instance or global.

Format of the variable-info object

Table 319 lists the fields in the variable-info JSON object.

Table 319. Get Archived Workflow Properties request: Format of the variable-info object

Field name	Type	Description
name	String	Name of the variable.
scope	String	Variable scope, which is either instance or global.
type	String	Type of variable, which is one of the following values: • boolean • string • number • date • time
value	String	Variable value.
visibility	String	Public or private.

Example HTTP interaction

In the following example, the GET method is used to retrieve information about an archived workflow. The workflow is uniquely identified by the workflow key, which is represented by the following string value: 2535b19e-a8c3-4a52-9d77-e30bb920f912.

```
GET /zosmf/workflow/rest/1.0/archivedworkflows/2535b19e-a8c3-4a52-9d77-e30bb920f912
HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 307. Sample request to get archived workflow properties

An example of the response is shown in Figure 308.

```
HTTP/1.1 200 OK
{
"percentComplete": 0,
"workflowDefinitionFileMD5Value":"6533178a155a98e784b056f0b16d060f",
"statusName": "archived",
"vendor": "IBM",
"accountInfo": null,
"workflowVersion": "1.0",
"automationStatus": null,
"access": "Public",
"productVersion": "Version 1",
"productID": "ABC123",
"category": "configuration",
"system": "DUMBPLEX.DUMBNODE (DUMBNODE)",
"workflowName": "Sample demonstrating variable substitution and the
use of a wizard. - Workflow_5",
"workflowDescription": "Sample demonstrating variable substitution and the
use of a wizard.",
"workflowID": "wizardSample",
"owner": "domadmin",
"jobStatement": null,
"workflowKey": "2535b19e-a8c3-4a52-9d77-e30bb920f912",
"productName": "Product ABC",
"archivedTime": "2016-10-11 09:28:33"
}
```

Figure 308. Sample response from a get archived workflow properties request

Delete an archived workflow

You can use this operation to remove an archived z/OSMF workflow from a z/OS system.

HTTP method and URI path

```
DELETE /zosmf/workflow/rest/<version>/archivedworkflows/<workflowKey>
```

In this request, the URI path variables are described, as follows:

- *<version>* identifies the version of the z/OSMF workflow service. The following value is valid: `1.0`.
- *<workflowKey>* identifies the archived workflow to be deleted.

Query parameters

None.

Description

This operation is used to delete an archived workflow from z/OSMF, including any notes that accompany the workflow and its steps, and the history log for the workflow.

Request content

None.

Authorization requirements

This request is available to the workflow owner only. A delete request from another user is rejected with the HTTP status code `403` (Forbidden) and an appropriate error message in the JSON response object.

For other authorization requirements, see "Authorization requirements" on page 525.

HTTP status codes

On successful completion, HTTP status code `204` (No content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code that is indicated and associated error message.

Table 320. HTTP error response codes for a delete archived workflow request

HTTP error status code	Description
`HTTP 400 Bad request`	The request contained an incorrect parameter, such as an incorrect workflow key.
`HTTP 403 Forbidden`	The requestor user ID is not permitted to delete the workflow properties.
`HTTP 404 Not found`	The specified workflow key was not found; the workflow does not exist.

Other standard status codes can be returned, as described in "HTTP status codes" on page 526.

Response content

None.

Example HTTP interaction

In the following example, the DELETE method is used to delete an archived workflow. The workflow is identified by the workflow key, which is the following string value: `7c4bac42-16a3-4af5-a5b9-263e60b280a4`.

```
DELETE /zosmf/workflow/rest/1.0/archivedworkflows/7c4bac42-16a3-4af5-a5b9-263e60b280a4 HTTP/1.1
Host: zosmf1.yourco.com
Connection: close
Authorization: Basic em9zbWZhZDp6b3NtZmFk
```

Figure 309. Sample request to delete an archived workflow

For a successful request, the HTTP response `204` is returned.

```
HTTP/1.1 204 No Content
```

Figure 310. Sample response from a delete archived workflow request

Chapter 2. Creating workflow definitions for z/OS

This information describes how to create workflow definitions that can be used with the Workflows task of z/OSMF. Included is an introduction to workflows concepts and a description of the basic elements of a workflow definition.

What is a z/OSMF workflow?

Generally, a *workflow* guides you through the complete set of steps that are needed to accomplish a goal, and, when dependencies exist, controls the sequence for performing those steps. In this way, a workflow can help to ensure that the steps are performed in the correct order, and prerequisites and dependences are identified clearly along the way. Conceptually, a workflow encompasses both the work to be performed and its performers. By identifying the individual steps to be performed, a workflow allows for the steps to be divided among different areas of an organization, and different members of a team. Using a workflow, a project owner can delegate specific items to the team members best suited to carrying out particular tasks.

In z/OSMF, a workflow is a guided set of steps that help you perform an activity on z/OS, such as configuring a software product or component, managing a z/OS resource or structure, or simplifying some relatively complex operation. To support these activities, a workflow can be designed to perform a wide variety of operations, such as starting z/OS subsystems, submitting jobs and scripts, and invoking TSO/E functions in batch (assuming that the workflow user is properly authorized).

In short, a z/OSMF workflow:

- Is based on a structured set of steps that are designed by a *workflow author*.
- Is described to z/OSMF through a *workflow definition*. A z/OS organization can write its own workflow definitions or obtain definitions from a third-party source (a *workflow provider*). z/OSMF includes samples for the workflow authors to reference when they create workflow definitions.
- Is created when a user imports a workflow definition into the z/OSMF Workflows task.
- Identifies steps to be performed and allows for these steps to be divided among different areas of an organization, which helps to facilitate user activities on z/OS.
- Contains one or more steps that guide the user through some action to be performed. Steps might consist of manual instructions for performing the steps, or might include some form of guided assistance, such as submitting a batch job, running a REXX script or a shell script, or creating files, based on user inputs.

In z/OSMF, the Workflows task allows a z/OS installation to create and manage workflows for performing activities on the z/OS system. The user who is responsible for the workflow and ensuring that it gets completed is the *workflow owner*. The workflow owner assigns workflow steps to users, making them *assignees* of the step. The user who accepts ownership of a step becomes the *step owner*.

In the Workflows task:

- The Workflows page displays the existing workflows for an installation, and provides the control point for creating and managing workflows.
- The Steps page displays the steps in a workflow, and provides the control point for managing the steps. From this page, you can select actions for the steps, such as assigning steps, changing ownership of steps, and performing steps.

The following topics provide more details on workflows, workflow steps, and the process of creating workflow definition files for use with the Workflows task of z/OSMF:

- "Terms you should know" on page 592

- "The Workflows task schema" on page 594
- "Creating the workflow definition file" on page 594
- "Defining steps for your workflow" on page 608
- "Workflow XML reference" on page 643.

Terms you should know

Workflow authors should be familiar with the following terms.

Workflow

1. An activity that is associated with the z/OS system, such as configuring a component or product. **2.** The instantiation of a workflow in z/OSMF, based on a workflow definition. A workflow consists of one or more units of work to be performed on the z/OS system, as described by the workflow definition. A workflow is created when the Workflows task is used to create an instance of a workflow from a supplied workflow definition file.

Workflows task

The task in the z/OSMF navigation area that allows users to interact with workflows on z/OS.

Workflow category

A classification of the activities to be performed in the workflow. In z/OSMF, a workflow that is used to configure system software is classified as a *Configuration* workflow. A workflow that is used to provision system software is classified as a *Provisioning* workflow. All other workflows are classified as *General* workflows.

Workflow definition

The logical structure of a workflow, represented as a series of one or more steps. The workflow definition identifies the various system objects and actions that constitute activities on z/OS and the rules for performing those activities. The workflow definition includes all of the information that is specified in, or referenced by, the primary XML file (the *workflow definition file*) and possibly other files that are included by the workflow definition file. This content typically includes information about the workflow (such as name and version), step definitions, variable definitions, file templates, and bundle files.

Workflow definition file

The primary XML file for a workflow definition. A workflow is stored in z/OSMF when its workflow definition file, and optionally, a workflow variable input file, is imported into the Workflows task.

Workflow variable input file

An optional file that supplies default values for one or more of the input variables that are defined in the workflow definition file. The workflow variable input file is specified as input when the workflow definition is imported into the Workflows task. Typically, a workflow provider might supply a workflow variable input file to save users from having to manually enter inputs when they perform a workflow.

Output file

A file that is created by a step, as the result of running a program. Typically, the file holds the results of a batch job, shell script, or REXX exec program, as determined by the workflow author. The output file can be used by other steps or workflow instances. In practice, a step might submit a batch job to create some z/OS related parameters, which are then used by a subsequent step, thus saving the Workflows task user from having to enter the parameter values manually.

Global variable

A variable definition that is available to all workflow instances. The Workflows task saves global variables in a repository that is called the global variables pool.

Instance variable

A variable definition that is available only to instances of a particular workflow.

Predefined variable
A variable definition that can be used for string substitution in the current step only.

Workflow author
The person, typically a programmer, who creates the workflow definition by using the XML tagging language.

Workflow owner
The user who is given ownership of the workflow through the Workflows task. The workflow owner is responsible for delegating the work in the workflow to users to perform (the step assignees).

Workflow provider
The source of the workflow definition file, typically IBM, or a software vendor.

Step A single, logical unit of work in a workflow. Consider each step to describe a specific activity to be performed on the system. A step is available to be performed when the workflow owner assigns the step to a user through the Workflows task, and the user accepts ownership of the step.

Step owner
The user who accepts ownership of a step and therefore responsibility for performing the step.

Automation processing
The processing of a workflow that contains one or more automated steps. A workflow that is comprised entirely of automated steps can complete with little or no user intervention. When automation processing is started on the workflow, the workflow runs to completion or until it is stopped by another condition, such as a user request or an error.

Automated step
A step can be designed to run automatically (without user interaction) when it is in Ready state. Such a step is referred to as an *automated step*. A workflow that is comprised entirely of automated steps can complete with little or no user intervention.

Batch execution step
A template step that runs an executable program as a batch job, such as a JCL job, a REXX exec, or a UNIX shell script. Contrast with an *immediate execution step*, which is a template step that runs a program in real time.

Conditional step
A step that can be performed when a logical condition is satisfied on the z/OS system or in the Workflows task. For example, a conditional step might become eligible to be performed if a job that is run by another step ends with a particular return code. A conditional step remains unavailable to be performed as long as the condition is not satisfied.

Called workflow
A workflow that is started by another workflow for execution. Conceptually, a called workflow is a step in the workflow that calls it (the calling workflow).

Feedback step
A step that includes a feedback form with questions for the step owner to answer at the conclusion of a step.

Immediate execution step
A template step that runs an executable program in real time, such as a REXX exec or UNIX shell scrip. Contrast with a *batch execution step*, which is a template step that runs a program as a batch job.

REST step
A step that issues a REST API request, such as a GET or PUT request.

Template step
A step that is designed to run an executable program, such as a JCL job, a REXX exec, or a UNIX

shell script. On completion, the results can be made available to other steps, in the form of variables or an output property file. Depending on how the program is processed, a template step is either of the following:

- *Immediate execution step*, which runs a program in real time
- *Batch execution step*, which runs a program as a batch job.

The Workflows task schema

A valid workflow definition file is one that follows the XML syntax and also conforms to the rules of the Workflows task schema.

The Workflows task schema is supplied with z/OSMF in the following location:

```
/usr/lpp/zosmf/V2R1/workflow/schemas/workflow_v1.xsd
```

The schema file is UTF-8 encoded.

If you are developing a workflow definition file, you require access to the schema, and therefore access to the z/OS system that is running z/OSMF.

Creating the workflow definition file

This topic describes the elements that comprise the workflow definition file.

A workflow is defined through a workflow definition file, which consists of one or more XML files and other types of files. Depending on the workflow design, a workflow might consist of just a single workflow definition file, or it might have a primary XML file that references one or more subordinate XML files, XML fragments, and external files. This document uses the term *workflow definition file* to refer collectively to all of the files that define a given workflow.

As a workflow author, you can create a workflow definition file in XML, in accordance with the schema that is supplied with the Workflows task of z/OSMF. The schema defines the required and optional properties (XML elements and attributes) of a workflow and imposes constraints on the order in which the elements are specified, and on the values that can be specified for each element and attribute.

It is assumed that workflow authors are familiar with the XML specification and coding practices. The following references provide additional helpful information:

- The World Wide Web Consortium (W3C) XML Technology page: http://www.w3.org/standards/xml/
- XML Core Working Group Public Page: http://www.w3.org/XML/

Besides XML files, a workflow definition might include external files. That is, apart from XML fragments, the workflow definition can refer to translated text files and velocity template files. These files must be read-accessible by the user creating (importing) the workflow.

You can provide the workflow definition file and any associated files in either a z/OS UNIX file or a z/OS data set. For a z/OS data set, use a sequential data set or a member of a partitioned data set (PDS).

Workflow Editor task in z/OSMF

To help you with creating and editing a workflow definition, z/OSMF includes an editor for workflows. You can use the Workflow Editor task to view, create, and modify workflow definitions. The Workflow Editor provides a visual framework for working with the elements of a workflow definition—the steps, variables, and workflow metadata.

The Workflow Editor task:

- Presents the details of a workflow definition in a graphical user interface (GUI).

- Provides easy-to-use options for viewing, creating, and modifying a workflow definition.

Using the Workflow Editor task, you can:

- Select an existing workflow definition file for editing
- View details about the different sections of a workflow definition
- Modify the workflow information, steps, and variables sections of the definition, including adding and deleting steps and variable definitions
- Overwrite the workflow definition with your changes.

To get started with the Workflow Editor task, in the navigation area, select **Workflow Editor**. More information about the Workflow Editor is provided in the online help.

Structure of a workflow definition file

Structurally, a workflow definition file is comprised of several sections, as follows:

- **Document declaration statements** that are not directly related to workflow content. These statements are required at the beginning of every workflow XML file, and are described as follows:

  `XML processing instruction`
  The primary XML file must start with a processing instruction (in column 1 of line 1) for the XML processor. This instruction defines the version of XML used and the encoding of the file. For example:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  ```

  `Document type definition (DTD)`
  You can optionally use a DTD to define entities (variables) in the workflow. Using a DTD with workflows is optional because z/OSMF uses the default XML schema to validate the contents of a workflow file, rather than the DTD.

  `Workflow root element`
  The workflow root element is the container for the main content of a workflow.
- **Workflow metadata**, which contains information about the workflow itself. For more information, see "Specifying the workflow metadata" on page 600.
- Optionally, a **manifest** of external files that contain translated text for the various user interface elements (widgets) that are displayed for the workflow. The Workflows task of z/OSMF can display text for widgets in the language defined for the browser locale, if you supply the translated text in a properties-based resource bundle file. By default, Workflows task displays the text in whatever language is used in the workflow file. Thus, it is recommended that you use the default language (such as English) in the XML and use bundle files to include any other supported languages. For more information, see "Including a manifest of translated text" on page 602
- Optionally, one or more **variable definitions**, which you can use to have the Workflows task prompt the user for input values. A number of data types are supported for variables, including string, integer, decimal, boolean, time, and date. A declared variable can be referenced by one or more steps in a given workflow. For more information, see "Defining variables for your workflow" on page 631.
- One or more **step elements** that describe the steps of the workflow. A workflow definition file must include at least one step, and should include all of the steps needed to complete an activity on z/OS (the workflow). For more information, see "Defining steps for your workflow" on page 608.

The reference tables in "Workflow XML reference" on page 643 summarize the basic elements of a workflow definition file, including the attribute values, descriptions, any default values, the XML attribute data types, and whether a particular attribute is required.

Creating and viewing the workflow definition file

This topic is intended to give application programmers guidance on how to create a workflow definition file.

To be considered valid, a workflow definition file must follow normal XML syntax conventions and also conform to the rules of the Workflows task schema that is supplied with z/OSMF.

Editing XML files on your workstation

It is recommended that you create and view the workflow XML files on a workstation, rather than on a z/OS system.

When working with XML files, use a text editor that includes an XML validation function. Validation is the process of comparing your XML files with the Workflows task schema. Doing so ensures that the files use only those tags that have been defined in the schema, and ensures that the files conform to the element rules specified in the schema.

To perform the XML validation, you will need to transfer the schema file to the XML editor on your workstation. The Workflows task schema resides on the z/OS system according to the z/OSMF product directory path. By default, at this location:

```
/usr/lpp/zosmf/V2R1/workflow/schemas/workflow_v1.xsd
```

For information about transport protocols, see "Transferring the workstation files to z/OS."

Specifying the processing instruction in the primary XML file

As mentioned in "Structure of a workflow definition file" on page 595, the primary XML file must begin with a processing instruction in column 1 of line 1. This instruction indicates to the XML processor the version of XML used and the file encoding format. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The following encoding formats are valid:

- UTF-8 (ASCII) or IBM-1047 (EBCDIC) for workflow definition files in UNIX files
- IBM-1047 for workflow definition files in z/OS data sets.

You must include a processing instruction in the primary XML file, but you do not need to specify a processing instruction in any XML fragment files that you include with the primary XML file.

Some workstation XML editors might not recognize the IBM-1047 processing instruction and will therefore not display the file. Also, it is not possible to display a file in EBCDIC on your workstation. As a workaround, if you want a file to be IBM-1047, you can specify UTF-8 in the processing instruction while you are editing the file on your workstation. Then, after transferring the file to z/OS in ASCII mode, you can edit the file on z/OS to change the processing instruction to IBM-1047, which allows the Workflows task to process the file correctly.

Transferring the workstation files to z/OS

It is recommended that you use File Transfer Protocol (FTP) to transfer the XML files to a z/OS system. Doing so helps to ensure that the files are encoded properly for use on z/OS.

For XML files:

- If the processing instruction (the first line in the XML file) indicates that the file uses UTF-8 encoding, transfer the file to z/OS in binary mode, to preserve the file encoding.
- If the processing instruction indicates that the file used IBM-1047 encoding, transfer the file to z/OS in ASCII mode, to have the file converted to EBCDIC.

As supplied by IBM, the Workflows task schema file is encoded in UTF-8. Thus, you should specify binary mode when transferring this file to a z/OS system. After transferring the files to the z/OS system, check the file permissions of the transferred files to ensure that they can be opened by the z/OSMF Workflows task. For testing and workflow development purposes, consider setting the file permission of the transferred files to the octal value of `0777` (in "properties").

Sample XML files for your reference

To help demonstrate various capabilities of the workflow XML schema, z/OSMF includes a number of sample XML files. It is recommended that you load these samples and observe their behavior in the Workflows task as you read this information.

The samples are supplied with z/OSMF in the `/samples` subdirectory of the product file system, which is, by default: `/usr/lpp/zosmf/V2R1/samples`.

Start with the following samples, which show a basic workflow definition, and demonstrate the use of language bundles and variables:

workflow_sample_basic.xml
: Shows the most basic workflow. It contains one step with only the required elements.

workflow_sample_translation.xml
: Shows a basic workflow that refers to a language bundle file. This workflow is used with `workflow_sample_bundle0.txt`.

workflow_sample_variables.xml
: Shows the use of variables that require user input.

More advanced concepts are illustrated in the following samples:

workflow_sample_automation.xml
: Shows the use of automated steps in a workflow.

workflow_sample_automation_property.txt
: Shows a sample workflow variable input file, for use with `workflow_sample_automation.xml`.

workflow_sample_output.xml
: Shows an example of writing generated variables to an output file.

workflow_sample_condition.xml
: Shows the use of conditional steps in a workflow.

workflow_sample_feedback.xml
: Shows an example of a feedback form that can be used to gather input on a step from the step owner.

workflow_sample_program_execution.xml
: Shows an example of running an inline executable program (a UNIX shell script) from within a step.

workflow_sample_rexx_template0.txt
: Shows how to invoke a REXX exec from a workflow.

workflow_sample_substeps.xml
: Shows the use of substeps and the use of step prerequisites to establish dependency chains.

workflow_sample_wizards.xml
: Shows the use of instructions and wizards that use input variables.

workflow_sample_file_template0.xml
: Shows the use of a file creation template.

workflow_sample_fragment0.xml
: Contains a sample step definition.

workflow_sample_fragment1.xml
: Contains a sample step definition.

workflow_sample_include_external.xml
: Shows the use of a DTD to make references to external files. This sample also demonstrates other features of steps, and uses some HTML tags within a step description.

References to external files

When you refer to an external file from the primary XML file, observe the following considerations:

- If the external file resides in a z/OS data set, specify the fully qualified data set name, which is preceded by a double forward slash (//). Do not enclose the data set name in single quotation marks.

  For example:

  ```
  //SYS1.PRODUCTX.TESTFLOW
  //SYS1.PRODUCTX(TESTFLOW)
  ```

- If the external file resides in a UNIX file, you can specify an absolute (fully qualified) path name, or a relative path name (that is, relative to the primary XML file).

When the workflow definition file is imported into z/OSMF, the Workflows task verifies that each referenced file exists and that the user has READ access to the files. The Workflows task then makes copies of the files, and later refers only to the copies.

Using variable substitution in the workflow definition file path

If you refer to an external file in your workflow definition, you can use variables in the file path name. Doing so allows users of the workflow to customize the file path for their environment. Thus, a file path that uses variables can add flexibility to your workflow definition.

To enable a workflow definition for file path substitution, on the <fileTemplate> element, set the attribute filePathSubstitution to true. Doing so means that the workflow user is responsible for ensuring that any variables that are used in the file template path must be replaced with valid values.

To supply valid values, the workflow user must edit the workflow input property file and replace the substitution variables with installation-specific values. The user must do this substitution before creating the workflow in the UI. The values that are supplied for the variables in file path are used only during workflow creation time, and cannot be changed during the workflow.

The default value for "filePathSubstitution" is false.

For example, assume that your workflow definition is defined, as follows:

```
<fileTemplate substitution="true" filePathSubstitution="true">
/u/${instance-filepath}MyTemplate.txt
</fileTemplate>
```

Here, the workflow user must provide a value for the instance variable "filePath" in the input property file, such as: `filePath=testDir`.

When the user proceeds through the Create Workflow dialog, the Workflows task performs the variable substitution to derive the actual file path for the step: `/u/testDir/MyTemplate.txt`

If your workflow definition is a UNIX file, the <fileTemplate> file path must be a UNIX path. If your workflow definition is a PDS member, the <fileTemplate> file path must be a data set name.

The Workflows task performs validation checking of the file path. A valid file path is one of the following:

- An absolute UNIX path name
- A fully qualified data set name (sequential or PDS) path name (a fully qualified name starting with "//")
- A relative path name, which is relative to the main XML file container. This structure can be a UNIX directory or a PDS. For a PDS, a relative path is the name of a member within the PDS. You cannot specify a relative path when the container is a sequential data set.

Notes:

1. If you do not specify filePathSubstitution="true", the file template path is treated as a UNIX path even if it contains variables. Remember, UNIX systems support most special characters in directory names, such as "$", "{", "}".
2. On creation of the workflow, the substituted file path is saved as a property to its corresponding workflow step. It cannot be changed during the workflow, regardless of whether the variable is changed later.

Defining entities for a workflow

You can use the document type definition (DTD) of XML to define entities in the workflow definition file. Entities are variables that can be referenced within the file. The XML processor replaces the references with the values specified in the DTD. You might use an entity to define a value that is subject to change as you develop the workflow, and thus can be changed in one place to affect all references.

You can define entities either inline, in one of the workflow XML files, or as system entities, in which case the XML processor obtains the replacement text from an external file.

Observe the following coding considerations:

- An inline entity can be used like a macro instruction. That is, you can define text in one place that is frequently referenced throughout the workflow definition file.
- A system entity is useful if you want to split the workflow definition file into smaller chunks for manageability, or reuse portions across multiple workflow definition files. For example, for a set of steps that is shared across different workflows. A system entity file can reside in the UNIX file system or in a z/OS data set, and the path name format is as described in "References to external files" on page 598.

Note, however, that the path name must be expressed as relative (not absolute). A further restriction for DTD entities is that the referenced entity must reside in the same "container" as the main workflow XML file.

- For a UNIX file, the referenced entity must reside in the same directory or a subdirectory of the primary XML file
- For a PDS, the referenced entity must reside in the same PDS. Here, a *relative path name* is simply the name of the member within the PDS.

Do not use a sequential data set to store an entity file. Also, be aware that a workflow definition that is contained in a sequential data set cannot refer to external entity files.

An entity file in a PDS member must start with the XML processing instruction, as described in "Creating the workflow definition file" on page 594, with IBM-1047 specified as the encoding format. In fact, any entity file using IBM-1047, whether it comes from a data set or a UNIX file, must start with this processing instruction. For UTF-8 files, the instruction is optional.

The following example shows how entities can be defined in the DTD.

```
<!DOCTYPE workflow [<!ENTITY copyright "Copyright IBM Corp., 2013">
                    <!ENTITY step1 SYSTEM "step1.xml" >
                    <!ENTITY step2 SYSTEM "step2.xml" >
                    <!ENTITY step3 SYSTEM "step3.xml" >
                ]>
```

An entity can be referred to in the document using the following notation:

```
&copyright;
```

For another example, see the sample file **workflow_sample_include_external.xml**, which is supplied with z/OSMF in the `/samples` subdirectory of the product file system.

Specifying the workflow root element

Use the workflow root element (<workflow>) to specify the XMLSchema-instance namespace, and optionally, a schema location.

For example:

```
<workflow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="workflow_v1.xsd">
```

The workflow element must end with a closing element: </workflow>.

Between the starting and ending element is the body of the workflow definition.

The schema file location specified here is ignored by the Workflows task when it validates an imported workflow definition file. You might find it helpful, however, to use a language-sensitive text editor when creating your workflow definition file. The path you specify indicates the location (relative to the file being edited if expressed as a relative path name) of the schema. Such editors can provide immediate feedback if you violate a rule of the schema, and might provide type-ahead support as you enter elements and attributes.

Though you might develop your workflow primarily on your workstation, and even though your editor is not flagging any errors, you do not know for sure that you have created a valid workflow until after you have imported it into the Workflows task of z/OSMF. Observe the following coding considerations:

- Differences exist among the various implementations of XML schema validators. Thus, a workflow definition file that validates in your text editor might not validate when it is imported into the Workflows task.
- The Workflows task performs additional validation that is not enforced by the schema, for example, validating that any message or template files that are referenced in the workflow actually exist on the z/OS system.

Specifying the workflow metadata

Use the workflow metadata element (<workflowInfo>) to specify information about the workflow, such as the workflow identifier, description, version, and vendor, and possibly other details. The Workflows task displays the metadata information to users in the Workflows table and the Properties for workflow page. Users can use the metadata values for filtering or configuring the workflows view.

At a minimum, you must specify the following metadata elements for a workflow:

workflowID
A short, arbitrary value that identifies the workflow.

This element can include the optional attributes *scope* and *isCallable*. For information, see "Workflow scope" on page 601 and "Callable workflows" on page 601.

workflowDescription
A short description of the workflow.

When a workflow is imported into z/OSMF, the Workflows task uses the workflow description (specified in the <workflowDescription> tag) to create a default workflow name, which the user can change.

workflowVersion
The version of the workflow definition.

The Workflows task caches only the latest version of any imported workflow definition file. Therefore, to ensure that the most current version is used, you must update the version value whenever you change any portion of the workflow definition. This includes changes to the primary XML file or any sub-files or referenced files. For this reason, when you author a

workflow definition, you might want to complete the development phase on a workstation before you import the workflow definition into the Workflows task.

vendor
The name of the workflow provider.

Table 326 on page 645 describes the elements that make up the workflow metadata.

Figure 311 shows an example of the metadata XML tags for a workflow.

```
<workflowInfo>
 <workflowID scope="system" isCallable="system">SampleWorkflow</workflowID>
 <workflowDescription>A simple workflow</workflowDescription>
 <workflowVersion>1.0</workflowVersion>
 <vendor>XYZ, Inc.</vendor>
 <General/> <!-- This element is empty, and completely optional. -->
</workflowInfo>
```

Figure 311. General metadata for a workflow

Workflow scope

It is possible to have multiple instances of a workflow run at the same time, on the same system or sysplex. In some cases, however, you might need to restrict a workflow to one instance only, which can be useful for coordinating actions across a system or sysplex. In z/OSMF workflows, this concept is referred to as the *singleton scope* for the workflow.

To set the singleton scope for a workflow, specify the optional attribute *scope* on the workflow metadata element (`<workflowInfo>`). Specify the scope attribute with one of the following values:

system
A maximum of one instance of this workflow can exist on any one system in the sysplex.

sysplex
A maximum of one instance of this workflow can exist in the sysplex.

none No limit exists for the number of instances of this workflow. For a callable workflow, this setting means that a new instance is always created on the calling system.

Workflow scope has ramifications for callable workflows. You might, for example, determine that a new instance of the workflow should always be created in response to a calling workflow. Or, you might prefer to have an existing instance of a workflow be used to handle the request. For considerations, see "Coordinating workflow-to-workflow actions" on page 621.

Callable workflows

In z/OSMF, a workflow can invoke another workflow for processing. A workflow that invokes another workflow is the *calling workflow*. A workflow that is invoked by another workflow is the *called workflow*.

To indicate whether a workflow is eligible to be called by another workflow, specify the optional attribute *isCallable* on the workflow metadata element (`<workflowInfo>`). On the isCallable attribute, also indicate the callable *range* for the workflow, that is, whether the workflow can be called only within the same system, or whether it can be called from any system in the same sysplex. Specify the isCallable attribute with one of the following values:

system
This workflow can be called only by another workflow that is running on the same system.

sysplex
This workflow can be called by any workflow that is running in the same sysplex.

If you omit the isCallable attribute, the workflow cannot be called by another workflow.

More information about callable workflows is provided in "Calling steps" on page 621.

Workflow category

In z/OSMF, a workflow category is a classification of the activities that are to be performed in a particular workflow. Workflows that are used to configure system software such as a product or component are *Configuration* workflows. Workflows that are used to provision system software, such as DB2 or IMS, are *Provisioning* workflows. All other workflows are *General* workflows.

The category is specified by the workflow author in the workflow definition file. The workflow category is optional. If no category is specified for the workflow, the workflow belongs to the General category. Figure 311 on page 601 shows an example of how the workflow category is specified: `<General/>`.

For a Configuration workflow or a Provisioning workflow, the workflow metadata specifies product-specific details, such as the product name, product version, and product ID. The metadata for a Provisioning workflow also specifies the type of software to be provisioned (software type). The Workflows task presents this information to users in the Category Specific tab on the Properties for workflow page.

The following example shows the metadata for a Configuration workflow:

```
<workflowInfo>
        <workflowID>Productx-01</workflowID>
        <workflowDescription>Initial configuration of Product X</workflowDescription>
        <workflowVersion>1.0</workflowVersion>
        <vendor>IBM</vendor>
        <Configuration>
                <productID>abc123</productID>
                <productName>Product X</productName>
                <productVersion>V1R99</productVersion>
        </Configuration>
</workflowInfo>
```

In this example, the metadata provides product-specific fields in the `<workflowInfo>` element, as follows:

productID
Identifier of the product or component that is being configured through the workflow, such as the product identifier (PID) or function modification identifier (FMID).

productName
Name of the product or component that is being configured through the workflow.

productVersion
Version and release of the product or component that is configured through the workflow.

Including a manifest of translated text

Use the translated text files element (`<translatedTextFiles>`) to define the *message manifest* for a workflow. The message manifest contains one or more bundle definitions, each of which contains one or more language file definitions. A *bundle* is a logical grouping for a set of text elements used in the workflow.

To refer to these files, other text elements in the workflow definition can specify a bundle name and an identifier (or key) within the language files that contain the replacement text for that language. The

Workflows task maps the bundle name and language into the file defined for that language and displays the text in the language that is in effect for the browser, if a language file is provided.

The Workflows task uses only the browser language to locate the translation file; the country is ignored. For example, `en-US` and `en-UK` are two versions of English: American, and British, respectively. The Workflows task uses only the value "en" to locate the language file.

The following example shows a message manifest:

```
<translatedTextFiles>
      <bundle name="StepMessages">
            <language name="en" path="steps/english.txt"/>
            <language name="fr" path="steps/french.txt"/>
      </bundle>
      <bundle name="VariableLabels">
            <language name="en" path="vars/english.txt"/>
            <language name="fr" path="vars/french.txt"/>
      </bundle>
</translatedTextFiles>
```

For an example of referencing a bundle in a translatable element, see the sample file **workflow_sample_translation.xml**, which is supplied with z/OSMF in the `/samples` subdirectory of the product file system.

Table 330 on page 653 describes the elements that make up the message manifest.

Enabling a workflow definition file for future upgrades

Over the course of time, you might want to provide users with a revised version of your workflow, presumably with enhanced functions. This topic describes the tags that you can use to make your workflow definition upgradeable.

In z/OSMF, to *upgrade* a workflow means to create a new instance of an existing workflow, based on a new definition file. As the workflow author, you can design your workflow definition to be upgradeable to future versions of the definition.

During an upgrade operation, the Workflows task user upgrades an existing workflow to a new level of the workflow by using the action **Create New Based on Existing**, which is provided in the Workflows table. By upgrading the workflow, the user creates a new instance of the workflow, while retaining data from the existing workflow.

Creating an upgradeable workflow definition

Using a set of tags in the Workflows schema, you can add an upgrade capability to your workflow definition. The tags specify which previous versions of the workflow are supported for upgrades. Also, the rules for data mapping, which variables, steps and attributes can be copied forward, and so.

To provide upgrade option in a workflow definition, you must provide a workflow definition with same workflowID as the current (old) version of the workflow definition.

To add an upgrade capability to your workflow definition, you define an upgrade element (`<preserveOptions>`) with the attributes described in this topic.

Start by defining the `<preserveOptions>` element after all of the defined steps in the new workflow definition. On the `<preserveOptions>` element, you can specify the following elements and attributes:

- Version. This element specifies information about the workflow to be upgraded. It contains the following attributes:

  - Value. Version of the workflow definition that is supported for upgrading. You must specify at least one prior version of the workflow definition.
  - Type. Specify either patch or release, as follows:
    - **Patch** indicates that the upgrade to is intended to fix a defect in the prior version of the workflow. Based on this idea, the new workflow replaces, rather than coexists, with the prior version. Thus, the prior version is canceled when the upgrade is performed.
    - **Release** indicates that the upgrade creates a new release of the workflow. The prior version is retained, based on the assumption that some installations with multiple releases might choose to use more than one version of the workflow.
- Variable set (<variableSet>). This element specifies which variable values are to be copied from the existing workflow. On this element, you can optionally specify the attribute (<defaultChecked>). Setting it to true (the default) causes the option **Copy attribute values based on upgrade definition** in the Workflows task wizard Create New Based on Existing to be selected by default. If so, all of the variable values are copied to the upgraded workflow.
- Step set (<stepSet>). This element specifies the step values to be copied from the existing workflow. On this element, you can optionally specify the attribute (<defaultChecked>). Setting it to true (the default) causes the option **Copy step attributes based on upgrade definition** in the Workflows task wizard Create New Based on Existing to be selected by default. If so, all of the current workflow step attributes are copied to the upgraded workflow. The attributes include the step assignees, step owners, step states, notes, submitted job status, and the record of how variable conflicts, if any, were resolved.
- Workflow history (<workflowHistory>). This element specifies whether the workflow history is copied from the existing workflow to the new instance. On this element, you can optionally specify the attribute (<defaultChecked>). Setting it to true (the default), causes the option **Copy workflow history** in the Workflows task wizard Create New Based on Existing to be selected by default. Otherwise, the option is not selected.
- Workflow notes (<workflowNotes>). This element specifies whether workflow notes are copied from the existing workflow to the new instance. On this element, you can optionally specify the attribute (<defaultChecked>). Setting it to true (the default), causes the option **Copy workflow notes** in the Workflows task wizard Create New Based on Existing to be selected by default. Otherwise, the option is not selected.
- Include (<include>). Use this element to specify which steps or variables in the prior workflow definition file are copied to the new workflow. You can specify this element multiple times. Specify either a regular expression or a variable name.
- Exclude (<exclude>). Use this element to specify which steps or variables are excluded from the set that is generated by the <include> element.
- Upgrade notes (<upgradeNotes>). Use this element to provide the user with information about the upgraded workflow definition. For example, you might use this element to provide the user with details about using the workflow.

For descriptions and data types of these elements and attributes, see "Workflow upgrade elements summary" on page 650.

The <include> element in the variable set specifies which variables are to be copied, based on either a regular expression (regExp) or the variable name. The following example specifies that all of the existing steps that match the specified regular expression are to be copied to the same steps in the new workflow definition:

```
<include regExp="\w*">
```

To copy the value from the variable that is named *setting3* in an existing workflow to the same variable in the new workflow definition, you might code the following:

```
<include name="setting3"/>
```

To copy the value from the variable that is named *setting1* in an existing workflow to the variable named *setting2* in the new workflow definition, you might code the following:

```
<include name="setting1" mapTo="setting2"/>
```

If you specify the `<include>` element multiple times in a sequence, and a variable is included for multiple times after calculation by name and regular expression, the last `<include>` specification is the one that is used for the copy operation.

Collecting user feedback

It is possible to collect feedback from the users of a workflow. A workflow author can optionally include a feedback form on one or more steps with customized questions for the step owner to answer at the conclusion of a step. Such feedback can be useful for determining the effectiveness of a workflow design, or collecting user requirements for future enhancements to a workflow. Inclusion of a feedback form is optional; answering the questions in a feedback form can be optional or required, as determined by the workflow author.

The workflows XML schema includes elements to help workflow authors create questions for users. The Workflows task includes functions to allow the workflow owner to prompt users for feedback, and collect the responses into a consolidated document. When all of the required feedback is provided, the workflow owner can send the feedback to the workflow author for evaluation.

How feedback is collected

Collecting feedback from the users of a workflow involves the participation of the following roles:

Workflow author
Workflow author defines feedback questions in the workflow definition file, and designates the questions as optional or required.

Workflow owner
For a workflow that includes feedback questions, the workflow owner is responsible for collecting the feedback. From the Workflows table, the workflow owner can select Feedback to launch actions related to feedback. The workflow owner can display pages to see which steps require feedback, which steps have incomplete feedback, and options for notifying the step owners who need to complete feedback.

Step owner
For a step that includes a feedback form, the step owner is responsible for providing feedback by answering questions about the step. To answer feedback for a step, the step owner selects a step and selects the table action Feedback. Only the step owner can display the feedback page for a step. The Feedback action is disabled for steps that do not contain feedback questions and for others other than the step owner.

When all of the required feedback is provided by step owners, the workflow owner can save the accumulated feedback into a feedback file. On the Generate Feedback Summary page, the workflow owner can create a report of the feedback, which can be sent to the workflow vendor for evaluation.

Schema elements

As the workflow author, you define feedback questions and answers in the workflow definition file.

Example

Figure 312 on page 607 shows how to define each of the question types in a workflow definition file.

In the example:
- `itemOne` defines a multiple choice question
- `itemTwo` defines an either or choice
- `itemThree` defines a question that accepts a write-in response.

```
<!-- ================= feedback definitions ================= -->

<feedbackItem name="itemOne">
   <question>How difficult was this step?</question>
   <!--The user must select one answer from the available choices-->
   <answers>
      <singleSelect hasOtherAnswer="true">
        <lable value="difficult">Very difficult</lable>
   <lable value="moderate">Somewhat difficult</lable>
   <lable value="moderate">Neutral</lable>
   <lable value="moderate">Somewhat easy</lable>
         <lable value="easy">Very easy</lable>
      </singleSelect>
   </answers>
</feedbackItem>

<feedbackItem name="itemTwo">
   <question>What did you like about this step?</question>
   <!--The user can select more than one of the available choices-->
   <answers>
      <multipleSelect hasOtherAnswer="true">
       <lable value="simple">Ease of use</lable>
  <lable value="info">Instructions were helpful</lable>
            <lable value="useful">Performed a useful function</lable>
            <lable value="quick">Ran quickly</lable>
      </multipleSelect>
   </answers>
</feedbackItem>

<feedbackItem name="itemThree">
   <question>How would you describe your experience with this step?</question>
   <!--The user supplies a text response (up to 500 characters) -->
   <answers>
  <text/>
   </answers>
</feedbackItem>
```

Figure 312. You can define a variety of questions for step owners to answer.

The questions defined in Figure 312 can be referenced by the steps in your workflow. Figure 313 on page 608 shows how the questions defined earlier can be included in the step definitions. Notice the attribute `required` is included for questions that require an answer from the step owner.

```
<!-- ================= step with feedback questions ================= -->
<step name="StepOne" >
        <title>A step with feedback</title>
        <description>A step with feedback.</description>
        <!--On the feedback tag, the attribute "name" identifies the
            feedback question. The attribute "required" indicates whether
            a user response is required or optional. -->
  <feedback name="itemOne" required="true"/>
  <feedback name="itemTwo" required="true"/>
  <feedback name="itemThree" required="true"/>
        <instructions>This step has three questions for you to answer.
                      All of the questions require a response.</instructions>
        <weight>1</weight>
</step>
.
.
.
<!-- ================= Another step with feedback questions ================= -->
<step name="StepTwo" >
        <title>Another step with feedback</title>
        <description>Another step with feedback.</description>
        <!--On the feedback tag, the attribute "name" identifies the
            feedback question. To indicate that a user response is required,
            the attribute "required" is included and set to true. To indicate
            that a response is optional, you can set required to false or
            omit this attribute. -->
             <feedback name="itemOne" required="true"/>
  <feedback name="itemThree"/>
        <instructions>This step has two feedback items. itemOne is required,
                      and itemThree is optional.</instructions>
        <weight>1</weight>
</step>
```

Figure 313. How feedback questions can be included in the steps in your workflow.

Defining steps for your workflow

A workflow is composed of one or more units of work called *steps*. A workflow definition file must contain at least one step; each step can contain substeps. In the Workflows task, a wizard guides the user through a step, which can be either manual or automated. For a manual step, the wizard only displays the instructions required for the user to perform the step. For an automated step, the wizard uses a template to create a file or execute a REXX exec, UNIX shell script, or JCL job. Optionally, the wizard can perform substitution by referencing variable values.

Every step has a name and contains a title and description. Steps can contain substeps, which can also contain substeps, up to five levels of nesting. For example, Step 1 can contain Step 1.1, which can contain Step 1.1.1, which can contain Step 1.1.1.1, which can contain Step 1.1.1.1.1. The total aggregation of steps and their substeps across the entire workflow cannot exceed 500.

The step name must be unique across the entire workflow. The step name is not displayed in the Workflows task, but it is used within the workflow to reference prerequisite steps. The title should be brief, and will be displayed in the step table when opening a workflow. Step titles are indented for substeps. The step description can be more detailed. It is only displayed in the "General" tab when viewing the step properties to provide a bit more context, if necessary.

A step can be designated as optional. This designation has effects on how its weight is used in the calculation of the percentage-complete for a workflow. Weights are described in “Parent steps and leaf steps” on page 609.

Any step can optionally contain a set of references to prerequisite steps (by step name) which must be completed before this step can be performed. Each prerequisite is identified in its own <prereqStep> element. The Workflows task will display the prerequisite chain for a step (in the "Details" tab when viewing the step properties), and will keep track of dependencies in terms of the state of any given step.

Any prerequisite step must have been defined previously in the workflow (though not an ancestor of the referencing step). If the reference is to a step with substeps, the substeps are treated as prerequisites (they need not be explicitly specified). If only certain substeps under a parent step are dependencies, they can be listed explicitly.

A step can contain multiple prerequisites steps, but these need not be cumulative. That is, if Step3 depends on Step2, and Step2 depends on Step1, then Step2 would identify Step1 as a dependency, and Step3 would identify Step2 as a dependency. If Step3 identified both Step2 and Step1, no harm would occur, but it makes the workflow more complex than necessary, and perhaps more difficult to maintain over time.

A step can be defined as *conditional*. Such a step is available to be performed based on whether a logical condition is satisfied on the z/OS system. For example, a conditional step might become eligible to be performed if a job run by another step ends with a particular return code. A conditional step, which depends on a logical condition, is different than a dependent step, which depends on a particular step being completed. For more information, see “Making a step conditional” on page 628.

What is described here applies to any step. For the distinctions between a parent step and a leaf step, see “Parent steps and leaf steps.”

Table 331 on page 655 describes the elements that make up a step.

Parent steps and leaf steps

A *parent step* contains a set of nested step elements (at least one). A step with no substeps is called a *leaf step*.

A leaf step actually performs, or tells the user how to perform, the actions required to complete the step. Leaf steps contain, at a minimum, instructions and a weight. Optionally, a leaf step can also contain a skills category, a template (for file creation or job submission), and references to variables which can be substituted into the instructions or template, or both.

The skills element (<skills>) specifies a suggested skills category for performing the step, such as "Security administration" or "Network administration." The Workflows task displays this value in the step table for a workflow. This value is free-form; specify it at your discretion.

The weight element (<weight>) specifies the relative difficulty of the step (a positive integer value from 1 to 1000). The Workflows task uses this value in the calculation of the percentage-complete value that is displayed as the workflow is performed. The scale is arbitrary; specify it at your discretion. Specify a lesser value for a step in which the user performs a simple action, such as cutting and pasting some a command text that you provide. For a more complicated task, such as deploying a digital certificate on z/OS, specify a greater value.

The instructions element (<instructions>) defines the content of the *Review Instructions* tab of the Step Perform wizard in the Workflows task. In this element, you provide the detailed instructions on what the user must do to perform the step to completion. The instructions can contain certain HTML tags for formatting, and hyperlinks to refer to additional information. The instructions can also use variable substitution. The <instructions> element is required, and must contain some content. Otherwise the step cannot be performed (the Workflows task Perform tab is omitted for the step).

In the Workflows task, the *Review Instructions* tab is displayed after variable-prompting. Thus, instructions can contain substituted values. However, instructions cannot be used to guide the user through the variable gathering stage. The variable attributes (title, abstract, and description) must be sufficient to guide the user, though you can also provide some guidance or context in the step description.

Review Instructions is the final tab that the user sees when performing a manual step. The user is expected to follow the instructions and then press **Finish** to complete the step.

Usually, a step must be marked complete before the workflow can continue. Depending on your design, however, you might want to allow the user to mark a step as *Failed* manually. This option might be useful if a step cannot be performed manually (outside of the workflow). To enable this option, include the can-mark-as-failed element (`<canMarkAsFailed>`) for the step and set it to true. Doing so causes the *Review Instructions* tab to prompt the user to confirm whether the step could be completed manually. If appropriate, the user can mark the step as *Failed* and continue with the workflow. By default, this option is disabled (the can-mark-as-failed element is set to false).

Leaf steps can contain optional references to variables that were defined earlier in the workflow definition. Based on your requirements, you might need to allow the step owner to modify the value of a referenced variable. To do so, identify the variable reference with the variable value element (`<variableValue>`). If so, the Workflows task displays a wizard to guide the user through entering values for the variable when performing the step, as described in "Defining variables for your workflow" on page 631. If a step only references a variable for read and not for modification, you do not need to specify the `<variableValue>` element.

You can code symbols using the Velocity syntax in both the instructions and the template that can be replaced by the value of the variable as entered by the user. More information is provided in "Using Velocity templates for variable substitution and other functions" on page 631.

You can determine how and when your workflow should prompt the user for variable values. For instance, you could have a step that is used only to prompt for the variables, which are then referenced by subsequent steps. Or, you can prompt for the variable directly within the step that uses it.

When coding the variable reference within a step, you can specify whether the variable is required. If a variable is required, and the user does not enter a value for it, the Workflows task prevents the user from proceeding until a value is specified. If a variable is optional, and the user does not enter a value for it, the Workflows task allows the user to advance to the edit screen, and the Velocity symbol in the instructions or template is displayed as-is without substitution. You can, however, use conditional Velocity statements to generate different text based on whether a variable has a value. If the symbol is left unresolved, a user might not understand these references, depending on how you name the variable. An unresolved variable in a template will likely result in an incorrect file or executable. However, the user can optionally edit the files before saving them or running them.

The variable reference also specifies whether to allow the user to change the value if the variable has one already. As a workflow author, you might know that the variable has already been used for an important purpose, such as allocating a file, and that, if the user changes the value, the overall procedure defined in the workflow would be corrupted. Here, you should not allow the user the opportunity to change the value. Instead, the Workflows task displays the value in read-only mode. The user can still override the value, after responding to a confirmation prompt that includes a list of any other steps that reference the same variable. For any variable that is referenced by additional steps, the Workflows task displays a list of those steps, along with the variable description, when the user presses the information icon. Even if you make a variable read-only when it already has a value, the user can override the value by checking a box in the description to make the value editable, thus accepting the consequences of doing so.

Template steps

A step that runs a program, such as a JCL job, a REXX exec, or a UNIX shell script, is referred to as a *template step*. On step completion, the program results can be made available to other steps, in the form of variables or an output property file. This topic describes how to write a template step, so that you can run programs and batch jobs in your workflow.

To enable a step to run a program or batch job, you must include the template element (`<template>`) and related elements and attributes from the Workflows schema in the XML structure for the step.

For the program that you want to run, you must determine:

- Whether to include the program code inline (that is, within the step XML structure), or in a separate, external file that can be called from the step. You indicate your choice by using the appropriate element tag, as follows:

  `<inlineTemplate>`
  Program code is specified inline in the step XML.

  `<fileTemplate>`
  Program is contained in an external file. The path name is also included on this element tag.
- Whether the program uses variables. If so, include the "substitution=" attribute on the `<inlineTemplate>` or `<fileTemplate>` tag. For more information, see "Using variables in a template step" on page 612.
- Whether the program runs in real time or is submitted as a batch job. You can choose to:
  - Run a program in real time for immediate results. The program can be a REXX exec or a UNIX shell script.
  - Submit a job for batch processing. The batch job contains JCL and might also imbed an executable program that runs under batch, such as a REXX exec or UNIX shell script.

  Include the submit as element `<submitAs>` to indicate the type of program processing to be performed. The Workflows XML schema supports the use of an inline template or a file template for all of the program processing types. For more information about the element `<submitAs>` and program processing types, see "Running a program in real time" on page 612 and "Submitting a JCL job for batch processing" on page 614.
- Whether the program output is saved. For a step that saves its output, you can use the `<output>` element to name the output file that is produced by the step. The `<output>` element specifies a default data set name or UNIX path name for the output file.

  A step can use an output file to write variables for subsequent steps to reference, as described in "Saving properties to an output file " on page 640.

  In the Workflows task, the step owner can override the value of the `<output>` element and choose a different output file, if wanted.
- Whether the program execution results are saved. For information, see "Saving the results of a program or job" on page 615.

Note:

- Avoid creating an inline template with characters that interfere with XML. If this problem occurs, use a file template instead. Also, when you use Velocity comparison operators in the instructions, do not use the less than (<) or greater than (>) characters, as they interfere with XML. Instead, use the alternative notation: `lt`, `le`, `gt`, and `ge`.
- No white space manipulation occurs for the inline template. Use care when typing the template into the XML, based on the context of the template. If writing JCL, for example, start the first line of the template immediately after the beginning `<inlineTemplate>` element. Subsequent lines must start on column 1, rather than being indented for readability.

How the user interacts with a template step

When a step includes a template element, the Workflows task enables the **Next** button on the wizard instructions page. When the user presses **Next**, the wizard guides the user through the activity, such as running a program or job, or creating and saving an output file. The behavior of the wizard is further controlled through the sub-elements that you define within the template element.

Using variables in a template step

The "substitution=" attribute on the template element indicates whether the template contains variables. The default is `false`. If you specify "substitution=true," you must also specify at least one variable value element (`<variableValue>`) in the step. Otherwise, the workflow fails validation when the user attempts to import the workflow definition into the Workflows task.

Observe the following considerations:

- If you include more variables than you reference, no errors result.
- If you refer to variables that are not included, but you do include at least one variable, no validation errors result. However, the Workflows task cannot resolve the variables during the substitution process.

Running a program in real time

A template step that runs a program in real time is called an *immediate execution step*. To run an executable program (a REXX exec or UNIX shell script) in real time, you include one of the following attributes on the submit as element (`<submitAs>`):

TSO-REXX
: Run a REXX exec program in real time.

TSO-UNIX-REXX
: Run a REXX exec program for the UNIX environment in real time.

TSO-UNIX-shell
: Run a UNIX shell script in real time.

The program results are immediately available to the step owner.

When running a program using these program types, understand that the program must run in the same system as the workflow instance that contains the step.

On completion of program execution, standard TSO/E service messages are written to the script log. As the workflow author, you can supplement the TSO/E messages with your own customized messages to indicate successful completion or errors. To do so, add the following elements to the `<submitAs>`:

- `<successPattern>`*`success regular expression`*`.*</successPattern>`
- `<failedPattern>`*`failure regular expression`*`.*</failedPattern>`

The element `<successPattern>` is required. You must specify one (and only one) regular expression for a successful program completion. The element `<failedPattern>` is optional. You can omit this element or specify up to 100 different specifications for the element `<failedPattern>`.

The messages are limited in size, based on the program type, as follows:

TSO-REXX
: The maximum length is 10000 lines.

TSO-UNIX-REXX
: The maximum length is 255 characters; extra characters are truncated.

TSO-UNIX-shell
The maximum length is 255 characters; extra characters are truncated.

To enable an executable program to receive input parameters from the step owner, include the following elements on the `<scriptParameters>` element:

`<description>`
Text description of the parameter, such as its intended use or recommended value.

`<value>`
Value of the parameter.

To enable an executable program to create output variables for use elsewhere in the workflow, add the element `<outputVariablesPrefix>` to the `<submitAs>` element. On this element, you can specify a meaningful prefix that identifies a string as a variable.

To manage potential variable name conflicts, you can specify a default behavior by adding the attribute `<needResolveConflicts>` to the element `<outputVariablesPrefix>`. By default, the step owner is prompted to choose the appropriate variable value in the Workflows task.

The executable program runs in a TSO/E address space on the z/OS host system. You can control how the TSO/E address space is started by including the following elements:

`<regionSize>`
Region size (in kilobytes) to be used for the address space. The valid range of values is 50000 to 2096128 (kilobytes). The Workflows task allows the step owner to specify a different region size, or use the supplied value. If no value is specified, the region size is 50000 kilobytes, by default.

`<procName>`
TSO logon procedure to be used for the address space. If no value is specified, the default is to use IZUFPROC, which is supplied by IBM as a cataloged procedure in proclib.

You can use the element `<timeout>` to set a time limit for an executable program. The valid range of values is 60 to 3600 (seconds). If no value is specified, the timeout value is 1800 seconds (30 minutes), by default.

Before running the program, z/OSMF saves the program to a temporary file. On completion of the program execution, z/OSMF deletes the temporary file. The location of the temporary file depends on the program type, as follows:

TSO-REXX
`userID.IZUWFTSO.W`, plus a randomly generated 7-digit number.

TSO-UNIX-REXX
`/tmp/fileName-scriptRexxFile.rexx`, where *fileName* is a randomly generated 32-digit number

TSO-UNIX-shell
`/tmp/fileName-scriptShellFile.sh`, where *fileName* is a randomly generated 32-digit number.

Other related output files are saved to the following temporary locations:

- Variable output file is saved at `/data/app/IZUWorkflows-workflowKey/outputFiles/stepName-outputfile`
- Message output file is saved at `/data/app/IZUWorkflows-workflowKey/scriptFile/stepName-scriptOutputMessageFile`

For examples of running programs from a step, see file `workflow_sample_program_execution.xml`, which is supplied with z/OSMF in the `/samples` subdirectory of the product file system.

Submitting a JCL job for batch processing

A template step that runs a program as a batch job is called a *batch execution step*. The submit as element (<submitAs>) indicates the type of program to be run. To run an executable program as a batch job, such as a JCL job, a REXX exec, or a UNIX shell script, you include one of the following attributes on the submit as element (<submitAs>):

JCL Submit a JCL job for batch processing on z/OS. The results are indicated in the job log.

TSO-REXX-JCL
Submit a JCL job that contains a REXX program. The program runs as a batch job on z/OS; the results are indicated in the job log.

shell-JCL
Submit a JCL job that contains a UNIX shell script. The program runs as a batch job on z/OS; the results are indicated in the job log.

These program types are ultimately executed as a JCL job by the Workflows task, which creates the necessary JCL and JOB statement, and displays the job output in the Workflows task. The size of any program to be run (JCL, REXX, or shell) is limited to ten thousand (10000) lines of code.

In the following example, a template definition contains inline JCL that runs a TSO command:

```
<template>
        <inlineTemplate>//STEP1       EXEC  PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT    DD    SYSOUT=A
//SYSTSIN     DD    *
LISTUSER IBMUSER
/*
        </inlineTemplate>
        <submitAs>JCL</submitAs>
</template>
```

Suppose that the first line of the JCL were to be placed on the next line after the beginning <inlineTemplate> tag. Doing so would inject a space into the JCL stream and cause a JCL error. Similarly, if any of the lines of JCL were to be indented, a JCL error would occur. Therefore, unless the file or program is small, you should use <fileTemplate> instead, which identifies the path name of an external file that contains the template. For path name examples, see "References to external files" on page 598.

In the following example, the file that is named jcljob.txt contains the JCL and exists in the same UNIX directory as the workflow definition file:

```
<template>
        <fileTemplate>
                jcljob.txt
        </fileTemplate>
        <submitAs>JCL</submitAs>
</template>
```

The Workflows task routes the job to another system in the sysplex, as determined by the system name that is chosen by the user when importing the workflow. Both JES2 and JES3 are supported. The JOB statement is entered by the Workflows task user separately, and applied to the job stream before it is submitted. Do not include a job card in your JCL template.

Observe the following considerations:

- A JCL template is submitted by the Workflows task after it includes the user-specified JOB statement and the appropriate JES routing statement, if the job is to be run on a different system in the sysplex.
- A REXX template is run by an IKJEFT01 job. This job creates a temporary data set to contain the REXX exec, and then executes the exec from that data set.
- A shell script is run by the BPXBATCH program. A temporary directory and file is created to contain the script. The script is run from this location and then the temporary directory and file are deleted.

The <submitAs> element contains a "maxRc=" attribute, which identifies the maximum return code to consider as successful. When the user presses the **Close** button from the Status tab of the Perform page (which displays the job output), the Workflows task checks the job return code. If the return code is less than or equal to the maxRc value, the Workflows task marks the step as complete. Otherwise, the Workflows task marks the step as failed.

JCL limits the return code value to the range 0 – 4095. If you do not specify a return code, the Workflows task uses a value of zero by default. BPXBATCH, which runs a shell script, further limits the return code to the range 0 – 15. If you are familiar with BPXBATCH, you might be aware that this service multiplies the script return code by 256 to derive a final return code value. The Workflows task divides the result by 256. Thus, you can code your workflow to be consistent with the value returned by the script.

For a step that submits a job, you can use the <maxLrecl> element to specify the maximum record length, in bytes, for the input data for the job. This value is used when the step is performed automatically (<autoEnable> is set to true; see "Automated steps" on page 627). If the step is performed manually, the user can optionally specify the maximum record length on the Edit JCL page in the Workflows task. If you omit the <maxLrecl> element, the default is 1024. For a job that uses the IEBUPDTE program to create or modify data sets, set this value to the minimum value of 80 to ensure that fixed-length records of 80 bytes or less can be processed.

For examples of submitting batch jobs from a step, see file workflow_sample_wizards.xml, which is supplied with z/OSMF in the /samples subdirectory of the product file system.

Saving the results of a program or job

To specify where to save the results of the step, depending on whether the template is used for file-generation or program execution, use the element <saveAsDataset> or <saveAsUnixFile>.

For a file create operation, the Workflows task recognizes that the file might be a configuration file and might therefore be required to reside in a data set or a UNIX file. Thus, the presence of each element determines whether the user is presented with a window to save the file in the corresponding location. The file is always saved in EBCDIC (code page IBM-1047). The elements serve another purpose in that a value can be specified by the workflow author as a default location, and that value itself supports variable substitution. This value is initialized in the widget that is presented to the user, who can override it.

For example, if you require the file to be saved in a data set, you would include a <saveAsDataset> element, but not a <saveAsUnixFile> element. If you did not care in which data set the file was saved, you can provide an empty element: <saveAsDataset/>.

If you want to provide a default value in which you substitute a high-level qualifier that is entered by the user, you can specify the <saveAsDataset/> element like this:

```
<saveAsDataSet substitution="true">${instance-hlq}.MYPROD(CONFIG)</saveAsDataset>
```

You can provide either, both, or neither of these elements, with or without a default value, which might or might not have substitution, as needed. Providing neither element has the same result as defining both elements without a value. Consider it a shorthand way of indicating that you can tolerate the file in either a data set or a UNIX file, and you have no suggestion as to what to name the output.

For a program template, the Workflows task always presents the user with a window to save the generated job in a user-specified location. Here, you can use the <saveAsDataset> element or the <saveAsUnixFile> element to provide a default value, with or without substitution, for either or both elements.

Using variables in template steps

You can add customization capabilities to template steps through the use of step-specific variables and other techniques for designing steps, as described in this topic.

As the workflow author, you might want to:

- Specify certain variables as being able to be used in a JOB statement.
- Use the same JCL repeatedly over a number of steps, but with a different set of values substituted for each job submission.

This topic describes techniques that you can use to add flexibility to template steps, in the following topics:

- "Variable substitution in the JOB statement"
- "Predefined variables in a step" on page 617

Variable substitution in the JOB statement

As the workflow author, you might want to indicate that certain variables in the workflow can be used in a JOB statement. To make a variable available for user selection, include the exposeToUser (`<exposeToUser>`) element on the variable element tag. This element indicates that the variable is to be included among the user-selectable variables in the **List variables for substitution** window of the Workflows task.

With the exposeToUser element, you can identify a certain set of variables that may be used (through substitution) in the JOB statement when the step owner is performing the step. To use a variable, the step owner would have to modify the JOB statement to add the variable reference. The step owner can then view the substitution to see if the JOB statement is as desired. The step owner can optionally designate the edited JOB statement be used for all steps in the workflow instance, but must understand the impact of doing so.

Suppose, for example, that you want to allow the user to select a notification user ID for the job. The system will send a message to this user ID when the job completes.

You can use the following technique to present the user with a selectable user ID `FRED` that is intended to be used with the NOTIFY= keyword parameter in the JOB statement. On the variable definition, include the exposeToUser element:

```
<variable name="notify-user">
<label>Notify user ID</label>
<abstract>This user ID can be selected for job notifications</abstract>
<description>Simple string variable.</description>
<exposeToUser>
<usage>Use this variable with the NOTIFY= parameter of the JOB statement
so that its value will designate who is notified when the job completes.</usage>
</exposeToUser>
<category>exposeToUser variables</category>
<string>
<default>FRED</default>
</string>
</variable>
```

In the workflow step, when the user displays the Create JOB statement page, the default JOB statement for the workflow is shown. For example, the following JOB statement is supplied with z/OSMF:

```
//IZUWFJB JOB (ACCTINFO),CLASS=A,MSGCLASS=0,
               MSGLEVEL=(1,1),REGION=0M,NOTIFY=IBMUSER
```

On the Create JOB statement page is the option **List variables for substitution**. If the user clicks this option, the Workflows task displays the variables in the workflow that include the exposeToUser element. The user can make JOB statement substitutions by manually copying these variables into the JOB statement.

In this example, the `notify-user` variable would be displayed in the **List variables for substitution** window. To use the `notify-user` variable for the notification user ID, the user can copy it into the JOB statement, as follows:

```
//IZUWFJB JOB (ACCTINFO),CLASS=A,MSGCLASS=0,
              MSGLEVEL=(1,1),REGION=0M,NOTIFY=${instance-notify-user}
```

On selecting the **View JOB Statement Substitutions** option, the user can display the substituted values to verify that the JOB statement is correct. In this example, the notification user ID is resolved to the value `FRED`:

```
//IZUWFJB JOB (ACCTINFO),CLASS=A,MSGCLASS=0,
              MSGLEVEL=(1,1),REGION=0M,NOTIFY=FRED
```

Predefined variables in a step

Suppose that your workflow contains a number of steps that run the same job repeatedly, with the exception of some values in the body of the job that must change each time the job is submitted. As an alternative to maintaining a slightly different copy of the JCL for each step, you can use the same job and call it from multiple steps. This technique requires that each step supply its own substitution values or *predefined variables* for each job submission. Doing so allows you as the workflow author to adjust the contents of the job dynamically to suit each step, based on the predefined variables.

To specify a predefined variable for a step, include the predefined variable (`<predefinedVariable>`) element on the step element tag, with the following element attribute:

name=
: Name of the variable (a string).

A predefined variable is treated as a string substitution for the current step only. You can specify multiple predefined variables per step. To avoid overriding the variables defined for the workflow, use a unique name for the predefined variable.

REST steps

You can design a step to invoke a Representational State Transfer (REST) service. A step that calls a REST service is referred to as a *REST step*. This topic describes the use of REST steps and the Workflows schema elements that you can use to create them.

A typical use for a REST step would be to obtain values for your workflow and assign the values to variables defined in the workflow. Suppose, for example, that your workflow requires an IP address and a port number from a REST endpoint. The REST step can be used to invoke this REST endpoint and map the values from the REST endpoint response body to your IP address and port number variables.

An example of a REST step that obtains an IP address and port number is shown in Figure 314 on page 620.

Notes:

1. The REST service to be called must support the use of JSON as the content-type for the request body and the response body.
2. It is assumed that you understand the layout of the returned data and know where to obtain specific values from the JSON string for mapping to variables.
3. The returned values can be mapped to instance variables only. You cannot modify the values of global variables.

How the user interacts with a REST step

From the user's perspective, when a step definition includes the `<rest>` element, the Workflows task enables the **Perform** button on the step instructions page. When the user presses **Finish**, the perform wizard issues the REST request. The behavior of the wizard is further controlled through the sub-elements that you define within the `<rest>` element.

Elements of a REST step

In a workflow definition, a REST step is defined with the `<rest>` element and a number of sub-elements and attributes that provide details about the REST request. The details include the HTTP method, URI path, query parameters, request body, expected status code, and actual status code. The `<rest>` element also defines any mapping properties that you use to save the JSON properties (name and value pairs) in the response body to workflow variables.

Table 321 provides a summary of the REST step elements.

Table 321. Summary of REST step elements

Element Name	Description	Required or optional	Supports substitution	Default value
`httpMethod`	Indicates the HTTP method that is used for issuing the REST request. The following values are valid: • GET • PUT • POST • DELETE.	Required	No	None
`schemeName`	Scheme name that is associated with the REST request. If specified, this element must be set to "http."	Optional	No	Scheme name that is defined for z/OSMF, which is "http" by default.

Table 321. Summary of REST step elements (continued)

Element Name	Description	Required or optional	Supports substitution	Default value
`hostname`	Host name or IP address of the system to which the REST request is directed. For example: `www.ibm.com`.	Optional	Yes	Host name that is defined for z/OSMF, which is "*" by default.
`port`	Port number to use for the REST request.	Optional	Yes	Port that is defined for z/OSMF, which is "443" by default.
`uriPath`	URI path to use for the REST request.	Required	Yes	None
`queryParameters`	For a GET or POST request, this element contains the query parameters.	Optional	Yes	None
`requestBody`	For a PUT or POST request, this element contains the request body.	Optional	Yes	None
`expectedStatusCode`	The expected HTTP status code from the REST API request. If this value does not match the `actualStatusCode` value, the workflow step fails. This behavior is similar to what happens when a job template step returns a return code that is greater than the allowed maximum return code.	Required	No	None
`actualStatusCode`	The actual HTTP status code that is received from the REST request. To obtain this value, map it to a workflow variable.	Optional	No	None
`propertyMapping`	The property from the REST response body that is mapped to a workflow variable. You can specify multiple `propertyMapping` elements in a REST step.	Optional	No	None

Table notes:

1. The elements `<schemeName>`, `<hostname>`, and `<port>` are used together. You must specify all of these elements or none of them. Otherwise, the workflow fails validation when the user attempts to import the workflow definition into the Workflows task. If you specify none of these elements, the REST request uses the z/OSMF defaults instead.
2. You must include the "mapTo" attribute on the `<actualStatusCode>` element. Set it to the name of the workflow variable that is to receive the actual status code value from the REST request. For example:

   ```
   <actualStatusCode mapTo="status_code" />
   ```

Mapping the returned data to variables

You can assign the values that are returned from a GET request to variables that are defined in your workflow. To do so, include the `<propertyMapping>` element on the `<rest>` tag. This element identifies the required values in the JSON response body and saves the values to workflow variables. On the `<propertyMapping>` element, specify the value to be assigned on the "mapTo" attribute. You can specify multiple `<propertyMapping>` elements in a REST step.

In the following example, the workflow owner is defined in a property mapping.

```
<propertyMapping mapTo="workflow_owner">["name"]</propertyMapping>
```

If the response body is:

```
{ "name": "John" }
```

the workflow variable "workflow_owner" is assigned the value John.

Specifying substitution variables for REST step elements

You can specify substitution variables as values for the elements <hostname>, <port>, <uriPath>, <queryParameters>, and <requestBody>. Doing so allows you to set the values for these elements dynamically from the workflow variables.

For an element that contains variable substitution, do the following:

- Include the optional attribute "substitution" and set it to *true*. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution.
- Set the element value to *${instance-WORKFLOW_VARIABLE_NAME}*, where *WORKFLOW_VARIABLE_NAME* is the workflow variable that is defined in the workflow.

A sample REST step is shown in Figure 314. In this example, variable substitution is used in the <requestBody> element to allow dynamic substitution for the stackId and stackname properties.

```
<step name="get_ip_and_port_data">
        <title>Get values for workflow variables ip_addr and port from REST API call.</title>
        <description>Invoke RESTful API to get IP address and port number.</description>
 <prereqStep name="get_stack_data_for_rest_request"/>
        <instructions substitution="false"> Execute step to retrieve the IP address and port number
        from z/OS Communications Server.</instructions>
        <weight>10</weight>
        <skills>REST</skills>
  <rest>
  <httpMethod>POST</httpMethod>
  <uriPath>/zosmf/workflow/WorkflowManager/cloud/ipaddr/</uriPath>
  <requestBody substitution="true">
{
 "name" : "GGAIPA",
 "description" : "IP address created by GGA",
 "ipaddr" : "1.1.1.2",
 "usageType" : "Internal",
 "workloadDeploymentGroup" : "/wdg/12345",
 "deploymentId" : "deployerA",
 "recoveryMethod" : "MANUAL_DISRUPTIVE",
 "hostName" : "mvstst1",
 "stack" :
  {
   "stackId" : "${instance-stack_id}",
   "stackname" : "${instance-stack_name}"
  },
 "boundTcpPorts" : [
  "4",
  "81"
 ]
}
  </requestBody>
  <expectedStatusCode>201</expectedStatusCode>
  <actualStatusCode mapTo="status_code" />
  <propertyMapping mapTo="ip_address">["ipaddr"]</propertyMapping>
  <propertyMapping mapTo="port0">["boundPorts"][0]</propertyMapping>
  <propertyMapping mapTo="port1">["boundPorts"][1]</propertyMapping>
  </rest>
 </step>
```

Figure 314. Sample REST step definition with substitution variables and property mapping variables

In this example, several returned values are mapped to workflow variables, as follows:

- actualStatusCode is mapped to the "status_code" variable.
- "ip_address" property is mapped to the "ipaddr" variable from the JSON object in the response body.
- "port0" property is mapped to the first element in an array of ports in the JSON object in the response body.
- "port1" property is mapped to the second element in an array of ports in the JSON object in the response body.

Calling steps

Suppose that your workflow needs a function that is already performed by another workflow on your system. Rather than attempt to replicate the same function in your workflow, you can add a step to invoke the other workflow as needed. A workflow that invokes another workflow is the *calling workflow*. A workflow that is invoked by another workflow is the *called workflow*. On completion, the called workflow returns control to the calling workflow.

From the point of view of the calling workflow, the called workflow is simply another "step" to be performed. You define the called workflow in the workflow definition file for the calling workflow, on the step element (`<step>`), through a set of elements and attributes.

With the ability to call one workflow from another, you can follow a modular approach to designing workflows. You can include a common, frequently performed, action in one workflow and allow other workflows to call that workflow, as needed. By including one or more called workflows in your design, you can maximize reuse of your code by using "common workflows," and effectively, integrate several workflows together.

To make a workflow eligible to be called by another workflow, include the *isCallable* attribute on the workflow metadata element `<workflowInfo>`, as described in "Callable workflows" on page 601.

How the user interacts with a calling step

In the Workflows task user interface, a step that invokes a called workflow is indicated to the user with the following icon in the Workflow Steps page:

In the Workflows task, when the user reaches a step that calls another workflow, z/OSMF determines whether an instance of the called workflow is already active on the system. If so, the Workflows task redirects focus to the called workflow so that the step owner can complete it. If the called workflow is already completed, the Workflows task displays the Workflows Steps table, showing the step that invoked the called workflow as marked complete.

Otherwise, if the called workflow is not already created, the Workflows task opens to the Create Workflow page so that the user can create the workflow. The user can supply a name for the workflow on the Create Workflow page, or accept the default workflow name.

Coordinating workflow-to-workflow actions

The Workflows schema metadata attributes *scope* and *isCallable* can be useful for coordinating workflow-to-workflow actions across a system or sysplex. By setting these attributes in various combinations, you can achieve the following effects:

- Restrict the number of instances of a workflow to one instance (a *singleton*).
- Always attempt to use an existing instance of a workflow in response to a calling workflow. Or, always cause a new instance of a workflow to be created, even if an instance exists already.
- Limit the use of a callable workflow to the same system or sysplex.

The scope and isCallable attributes are specified on the workflow metadata element (`<workflowInfo>`), as described in "Callable workflows" on page 601.

Nine combinations of scope and isCallable are possible, as shown in Table 322 on page 622.

Table 322. Use scope and isCallable to coordinate workflow-to-workflow actions

scope value	isCallable value	Effect on the called workflow
system	system	Workflow is limited to one instance per system, and can be called on that system only. If an instance does not exist, an instance is created.
sysplex	system	Workflow is limited to one instance per sysplex, and can be called from the same system as the calling workflow.
none	system	A new instance is always created. The instance can be called from the same system as the calling workflow.
system	sysplex	Workflow is limited to one instance per system, and can be called from any system in the sysplex. If an instance does not exist, an instance is created. For an automated workflow, if the calling step is performed automatically, the called workflow is searched for only on the calling system. If an instance is found, it is used; otherwise a new instance is created on the calling system.
sysplex	sysplex	Workflow is limited to one instance per sysplex, and can be called from any system in the sysplex. If an instance does not exist, an instance is created.
none	sysplex	A new instance is always created. The instance can be called from any system in the sysplex.
system	none (omitted)	Workflow is not callable.
sysplex	none (omitted)	Workflow is not callable.
none	none (omitted)	Workflow is not callable.

How workflow access type is handled

When a workflow calls another workflow for processing, z/OSMF changes the access type for the called workflow to match the calling workflow. This processing ensures that the requested access type is applied consistently to both of the workflows in a calling relationship.

Note, however, that this processing is not performed when the called workflow is limited to one active instance in the system or sysplex.

Designing a step to call another workflow

To have a step call another workflow, specify the step element (`<step>`) with the elements for defining a called workflow. The major step elements for defining a called workflow in your workflow definition file are described in Table 323.

Table 323. Major step elements for defining a called workflow in your workflow definition file

Element name	Description	Required or optional
variableMapping	Used to transfer instance variable values between the called workflow and calling workflow, and specify options for sharing variables. More information about this element and its sub-elements is provided in "Sharing variables between the calling workflow and called workflow" on page 624.	Optional.

Table 323. Major step elements for defining a called workflow in your workflow definition file (continued)

Element name	Description	Required or optional
callingStepWeight	Specifies the relative difficulty of the step compared to other steps within this workflow (a positive integer value 1 – 1000). The Workflows task uses this value in the calculation of the percentage-complete value that is displayed for the calling workflow. The scale is arbitrary; specify it at your discretion. Consider the difficulty of the called workflow as single step among the other steps in the calling workflow.	Required.
callingStepSkills	Specifies a suggested skills category for performing the step, such as "Security administration" or "Network administration." The Workflows task displays this value in the step table for a workflow. This value is free-form; specify it at your discretion.	Optional.
callingStepAutoEnable	Indicates whether the step is to be performed automatically when all prerequisite steps are completed, and no user inputs are required. If *callingStepAutoEnable* is not specified, the default is *false*. More information about designing steps to run automatically is provided in "Automated steps" on page 627	Optional.
canCallingStepMarkAsFailed	Indicates whether the step can be marked as *Failed* manually by the step owner. When set to *true*, the Review Instructions page in the Step Perform wizard includes the option to allow the step owner to mark a step as *Failed* manually. When *false*, this option is not displayed to the user. If *canCallingStepMarkAsFailed* is not specified, the default is *false*.	Optional.
calledWorkflowDefinitionFile	Specifies the path name of an external file that contains the workflow definition for the called workflow. This element is optional; it is used only if z/OSMF must create a new instance of the called workflow on the system.	Optional.
calledWorkflowDescription	Specifies a short description of the called workflow. The Workflows task displays this text on the Create Workflow page, when it prompts the user for the workflow definition file.	Required.
calledWorkflowID	The name of the workflow to be called. The combination of calledWorkflowID and calledWorkflowVersion must be unique within the Workflows task.	A selection is required: Either calledWorkflowID or calledWorkflowMD5.
calledWorkflowVersion	The version of the definition file that is used to create the called workflow. The combination of calledWorkflowID and calledWorkflowVersion must be unique within the Workflows task. The Workflows task caches only the latest version of an imported workflow definition file. Therefore, to ensure that the most current version is used, you must update the version value whenever you modify any portion of the workflow definition file, including changes to any sub-files or referenced files. For this reason, when you create a workflow definition file, you might want to complete the development phase on a workstation before you import the workflow definition into the Workflows task.	Optional.
calledWorkflowMD5	An MD5 encrypted value (a 128-bit hash value) that you can use to identify the called workflow.	A selection is required: Either calledWorkflowID or calledWorkflowMD5.

Identifying the called workflow

To invoke a called workflow, the calling step must identify which workflow is to be called. The z/OSMF schema provides two methods for you to uniquely identify the called workflow. Use either of the following approaches:

- Specify the workflow ID of the called workflow on the workflow ID element (`<calledWorkflowID>`). You can further qualify this specification by optionally including the version of the workflow definition of the called workflow on the element (`<calledWorkflowVersion>`). The version is typically updated by the workflow author whenever any portion of the workflow definition file is changed.

The Workflows task caches only the latest version of an imported workflow definition file. Therefore, to ensure that the most current version is used, you must update the version value whenever you modify the workflow definition. For this reason, when you create a workflow definition file, you might want to complete the development phase on a workstation before you import the workflow definition into the Workflows task.

- Specify the called workflow MD5 element (`<calledWorkflowMD5>`). This element specifies a 128-bit hash value that can be used to identify the called workflow. You can specify this element in place of the workflow ID and version elements.

Note: No more than one level of nesting of called workflows is permitted in a workflow-to-workflow relationship. Thus, the specified workflow definition cannot contain a step that calls another workflow.

Sharing variables between the calling workflow and called workflow

It is possible to share variables between the calling workflow and the called workflow. Any variables that are defined to either workflow can be shared by using the element `<variableMapping>`.

This element consists of two sub-elements and their associated attributes, as follows:

- Use the element `<fromCallingToCalled>` to describe the variable values that are to be transferred from the calling workflow to the called workflow.
- Use the element `<fromCalledToCalling>` to describe the variable values that are to be transferred from the called workflow to the calling workflow. To handle variable conflicts, you can optionally include the attribute `override=` to specify whether the called workflow variables take precedence over the calling workflow variables. The default is `override=false`.

On each element, you can optionally specify the following sub-elements and attributes:

regExpression
Regular expression. Use this attribute to filter on variable names with one or more wildcard characters. For example, to select all variables prefixed with "setting," you can specify:

```
<regExpression>^setting.*$</regExpression>
```

variableName
Name of the variable. Use this element to identify the variable that is to be shared with the target workflow. The variable is also saved in the Workflows task variable pool.

To map this variable to a specific variable in the target workflow, include the attribute `mapTo=` on the element `variableName` and set it to the name to the target variable. The behavior of the attribute `mapTo=` on the element `variableName` depends on which element is used to pass variables, as follows:

- When specified on the element `<fromCallingToCalled>`, the variables are mapped to the target variables only when a new instance of the called workflow is created in response to the calling step.
- When specified on the element `<fromCalledToCalling>`, the variables are mapped to the target variables on completion of the called workflow.

Example of how a called workflow is defined in a step

As an example, assume that your workflow includes a step ("Define User"), which is used to define a user ID and security group to the system security product. Usually, to verify that this setup is done correctly, users would run another workflow. In this example, you add a step to invoke the other workflow directly as a called workflow. When the step owner selects this step to be performed, the Workflows task displays the called workflow, so that the step owner can perform it.

Further assume that a number of variables will be shared between the workflows through the use of the element `<variableMapping>`. The step that calls the workflow (the calling step) will pass a number of

variables on the element <fromCallingToCalled>. Similarly, the called workflow will pass a number of variables to the called workflow, on the element <fromCalledToCalling>.

The definition for a called workflow might be coded as shown in Figure 315 on page 626.

For illustrative purposes, the example in Figure 315 on page 626 shows a variety of methods for sharing variables between workflows, as follows:

- On the element <fromCallingToCalled>, the step that calls the workflow (the calling step) passes variables in the following ways:
  - To pass all variables with "setting" as the variable name prefix, the calling step specifies the element <regExpression>, as follows:

    ```
    <regExpression>^setting.*$</regExpression>
    ```

  - To pass the value of the variable called st_user to the called workflow, the calling step specifies the variable, as follows:

    ```
    <variableName>st_user</variableName>
    ```

  - If an instance of the called workflow is not already created, z/OSMF will create one in response to the called workflow. If so, the element <fromCallingToCalled> ensures that the new called workflow inherits the variables from the calling workflow. In this example, the value of the variable st_uid is passed to the calling workflow, and overlays the existing value of the variable named st_group because the attribute mapTo= is included on the element <variableName>, as follows:

    ```
    <variableName mapTo="st_group">st_uid</variableName>
    ```

- On the element <fromCalledToCalling>, the called workflow shares a number of variables with the calling workflow. Any variables to be copied back to the calling workflow are performed on completion of the called workflow. Here, the override attribute is included, so that the called workflow's variables will override those of the calling workflow:
  - To pass all variables with "set" as the variable name prefix, the element <regExpression> is specified, as follows:

    ```
    <regExpression>^set.*$</regExpression>
    ```

  - To pass the value of variable called st_uid to the calling workflow, and overlay its existing value for the variable named st_gid, the mapTo= attribute is included on the variableName element, as follows:

    ```
    <variableName mapTo = "st_gid">st_uid</variableName>
    ```

  - To pass the variable called st_user to the calling workflow, the variable is specified, as follows:

    ```
    <variableName>st_user</variableName>
    ```

```
<step name="Define User">
 <title>Ensure that the user ID and group are created.</title>

 <description>
   This step verifies that the required user ID and security
   group are created. This step invokes another workflow (a called workflow),
   which is identified here based on the workflow ID and version.
   Alternatively, we could have identified the called workflow using its
   MD5 hash value.
 </description>

 <variableMapping>
  <!-- Variables to share with the called workflow. -->
     <fromCallingToCalled>

        <!-- Use a regular expression to filter the variables. -->
         <regExpression>^settin.*$</regExpression>

         <!-- The following line copies the value of st_uid to the variable st_group. -->
       <variableName mapTo = "st_group">st_uid</variableName>

      <!-- The following line copies the value of st_user to the called workflow,
           if no st_user variable definition already exists in the called workflow.
           This value also will be saved in the Workflows task variable pool.-->
      <variableName>st_user</variableName>

     </fromCallingToCalled>

    <!-- Variables to share with the calling workflow. Here, the override attribute
         is set to true, so that the called workflow's variable values will override
         those of the calling workflow.  -->
      <fromCalledToCalling override= "true">
        <regExpression>^set.*$</regExpression>
        <variableName mapTo = "st_gid">st_uid</variableName>
        <variableName>st_user</variableName>
     </fromCalledToCalling>
 </variableMapping>

  <callingStepWeight>10</callingStepWeight>

  <callingStepSkills>System Programmer</callingStepSkills>

  <calledWorkflowDefinitionFile>\usr\lpp\zosmf\V2R1\samples\workflow_sample_wizards.xml
  </calledWorkflowDefinitionFile>

 <calledWorkflowDescription>This called workflow is used to help verify that the user
   and group are created successfully.</calledWorkflowDescription>

 <calledWorkflowID>workflow.sample.wizards</calledWorkflowID>

 <calledWorkflowVersion>1.0</calledWorkflowVersion>

</step>
```

Figure 315. Example: Defining a called workflow on the step element tag.

Automated steps

A workflow might have steps that can be performed without the need for user interaction, such as a job that can be submitted without user input. If so, you can designate the step as an *automated step* in the workflow definition file. Doing so instructs the Workflows task to run the step automatically, as soon as any prerequisite steps in the workflow are completed. By including one or more automated steps in a workflow, you help to simplify the user experience.

In the Workflows task user interface, automated steps are indicated to users in the following ways:

- In the *Workflow Steps* table, the column Automated indicates whether a step is automated, based on how the step is defined in the workflow definition.
- In the Workflows task main page, when an automated step is performed, the workflow status is indicted as *Automation in Progress*.

A workflow can consist of both automated steps and non-automated (manually performed) steps.

How the user interacts with an automated step

To perform an automated step, the user selects the **Perform** action in the *Workflow Steps* table, as is done for manual steps. For an automated step, the Workflows task presents the user with a dialog window to confirm whether the step and any subsequent automated steps should be performed automatically. Alternatively, the user can choose to perform the step manually, by selecting an option in the dialog window that is called Manually perform the selected step.

When started, an automated step—or series of automated steps—can run to completion or until stopped by another condition, such as a user stop request or a step error. Specifically, a workflow with automated steps can run until one of the following conditions occurs:

- Completion of all subsequent steps.
- Processing reaches an automated step for which one or more required variables are not satisfied.
- Processing reaches a non-automated step in the sequence of steps.
- Processing reaches an automated step that is not currently eligible for automatic processing. That is, the step is *Unassigned*, *Assigned*, *Not Ready*, or *Submitted*.
- Processing is stopped through a user request.
- An error is encountered.

A workflow that is comprised entirely of automated steps can run to completion without user intervention.

Tracking the progress of automated steps

For workflows that contain automated steps, z/OSMF creates notifications and history entries to inform step owners of the automation progress.

At the completion of an automated step or a sequence of automated steps, z/OSMF creates a notification to inform the step owner of the step status. If processing reaches a manual step that requires user interaction before the workflow can continue, z/OSMF creates a notification for the step owner to prompt for action. Similarly, if an automated step is stopped or fails for any reason, z/OSMF sends a notification to the step owner. In z/OSMF, users can access notifications through the Notifications task.

During the processing of an automated step, z/OSMF updates the workflow history to indicate the key checkpoints in the workflow progress, such as:

- Completion of the automated step
- Completion of all automated steps in the workflow
- Automation is started or stopped through user request (and by whom)
- An error is encountered during the processing of an automated step.

Automation progress is not displayed (in terms of step completion check marks) until the user refreshes the display.

Users of the Workflows task can view the details of the step status in the Workflow History table.

Design considerations for automated steps

Consider a step to be eligible for automation if it requires no user input at all, or if all of the required inputs can be supplied to the workflow at creation time, in the form of a workflow variable input file.

When you code the step element, you can specify whether the step is automated (that is, can be performed automatically by the Workflows) by including the `autoEnable` attribute on the step element (`<step>`). Set this attribute to true or false, as needed. By default, the `autoEnable` attribute is false. Figure 316 shows an example.

```
<step name="Sample_Automated_Step" optional="true">
<title>This is a very simple JCL job</title>
<description>This optional step submits an empty job using IEFBR14.</description>
<instructions>This step is performed automatically.</instructions>
<weight>1</weight>
<skills>Submit a job to run on z/OS</skills>
<autoEnable>true</autoEnable>
<template>
    <inlineTemplate>//STEP1      EXEC  PGM=IEFBR14
//SYSEXEC   DD   DUMMY
//* PRINT DD SYSOUT=A
/*
        </inlineTemplate>
        <submitAs>JCL</submitAs>
     </template>
</step>
```

Figure 316. You can designate a step as automated by adding the autoEnable element to the <step> element tag.

For any automated steps that you provide, it is recommended that you use the description tag to provide the user with enough information to understand the implications of allowing the step to run.

When automated steps are ordered consecutively in a workflow, a request to run the first automated step begins a process in which each subsequent automated step can run to completion, or until one of the steps encounters a condition that stops the processing of steps. For this reason, it is recommended that you group automated steps in the workflow definition file together, to take full advantage of this cascading behavior.

Making a step conditional

A *conditional step* is available to be performed when a logical condition is satisfied on the z/OS system or in the Workflows task. A conditional step might become *Ready* (eligible to be performed), for example, if a job run by another step ends with a particular return code. A conditional step remains *Not Ready* (unavailable to be performed) as long as the condition is not satisfied.

Understand that a conditional step, which depends on a logical condition, is different than a *dependent* step, which depends on a particular step being completed, to satisfy a prerequisite.

In the Workflows task user interface, conditional steps:

- Are indicated to users in the **Details** tab on the *Step Properties* page.
- Are shown in the *Not Ready* state until the condition is true (satisfied) — even when the prerequisite steps, if any, are complete.

A conditional step becomes ready for performing only when a specific condition is satisfied in the current step or a preceding step. Thus, the expression being tested and a text description are required sub-elements of the condition element.

Target states

Optionally, you can specify a desired *target state* for a conditional step. The target state specifies the state the step is to assume when the condition is true. Typically, the target state is *Ready*, which is the default value, if you choose to omit this sub-element.

The following target states are valid:
- Ready
- Skipped.

Types of conditional expressions

The following types of conditional expressions are supported:
- Expressions using logical operators AND (&) and OR (|). For example:

```
${step1.returnCode} == "0000" || (${step2.returnCode} == "0000" && ${step2.stepOwner} == "IBMUSER")
```

- Expressions based on ternary operators. For example:

```
   condition ? value_if_true : value_if_false
```

- Mathematical functions. For example:

```
Math.max(${step1.returnCode} , ${step2.returnCode} ) > 0
```

Design considerations for conditional steps

Observe the following design considerations for conditional steps:
- A conditional step must be a leaf step (a step with no substeps). A parent step cannot be a conditional step.
- When coding the step element, specify whether the step is conditional by including the `condition` attribute on the step element (`<step>`). Also, specify both the expression being tested for (typically a mathematical or logical expression) and a text description of the condition. Both the expression and its description are displayed to the end user in the **Details** tab on the *Step Properties* page.
- A conditional expression can refer only to preceding steps in the workflow.
- You can include workflow input variables in conditional expressions. Doing so allows conditional steps to resolve to true or false, based on installation-specific conditions.
- You can use the following step attributes in conditional expressions: `<stepState>` and `<returnCode>`.
- `<returnCode>` is a string type attribute; you cannot use it in a mathematical comparison. To compare a return code with a second return code or another numerical value, such as zero (0), you can write the condition expression like this: `${step2.returnCode} ) > "0000"`. Represent the return code string with four characters, for example `"0000"` or `"0008"`.

Example

As an example, assume that Step 3 should not be performed unless Step 1 and Step 2 complete with a return code zero. Here, the XML for Step 3 could be coded as follows:

```
<Step name=Step3">
 <title>A conditional step based on return code</title>
 <description>This conditional step is not ready until
         the two preceding steps complete with RC 0
 </description>
 <instructions>Run this job.</instructions>
 <condition>
   <expression>${step1.returnCode} == "0000" || (${step2.returnCode} == "0000"
   </expression>
  <description>This step requires that Steps 1 and 2 have
          completed successfully.
 </description>
 </condition>
 <targetState>Ready</targetState>
```

Figure 317. You can designate a step as conditional by adding the condition element to the <step> element tag.

The previous example can be expanded to include a condition, based on a variable value. In Figure 318, Step 3 is not performed unless Step 1 and Step 2 complete with a return code zero and the variable `${instance-st_user}` is IBMUSER.

```
<Step name=Step3">
 <title>A conditional step based on return code and user ID</title>
 <description>This conditional step is not ready unless
         the two preceding steps complete with RC 0
               variable st_user value is IBMUSER
 </description>
 <instructions>Run this job.</instructions>
 <condition>
   <expression><![CDATA[${step1.returnCode} == "0000" &&
                  ${step2.returnCode} == "0000" &&
                  ${instance-st_user} == "IBMUSER" ]]>
     </expression>
  <description>This conditional step is not ready unless the two
                preceding steps complete with RC 0 and the variable
                st_user value is IBMUSER.</description>
 </condition>
 <targetState>Ready</targetState>

⋮
</step>
```

Figure 318. You can use variable values in the condition to be satisfied.

Note that a variable reference can contain an underscore, for example: `${instance_st_user} == "IBMUSER"` or a hyphen, for example: `${instance-st_user} == "IBMUSER"`.

Using translatable strings

The tables in "Workflow XML reference" on page 643 indicate which elements have translatable values with a type of nlsString or nlsRichString.

A translatable string takes two optional attributes, bundle= and bundleKey=. If one attribute is specified, the other must also be specified.

Table 324 describes shows the bundle= and bundleKey= attributes.

Table 324. Translatable strings

Attribute name	Description	Type	Requirements and restrictions
bundle=	The name of a bundle defined in the message manifest	A single-token string	The referenced bundle must exist.
bundleKey=	The key within a language file containing the replacement text for the text element.	nonNullString	The referenced file should contain a key of this name, but this is not validated by the Workflows task.

Using rich translatable strings

Within the schema, any string defined with type nlsRichString or nlsRichVelocityString is a translatable string that can contain HTML tags. Not all HTML tags (and their attributes) are supported, though tags for headings, tables, lists, hyperlinks, and text formatting are available.

The allowable tags are: h1, h2, h3, h4, h5, h6, ol, ul, dl, dt, dd, li, br, p, hr, table, th, td (with the frame, rules, and width attributes), tr, caption, colgroup, col, thead, tbody, tfoot, i, b, u, em, strong, cite, code, samp, kbd, pre, tt, sub, sup, big, small

To specify a hyperlink, use the anchor (<a>) tag. When clicked, the hyperlink opens a new tab or window, based on the user's browser settings. You can specify the href attribute only. To include an ampersand character (&) in the URL, enclose the symbol in quotes: "&". Also, include the protocol with the URL. For example, a value of "http://www.ibm.com" is correct, but "www.ibm.com" is not.

Defining variables for your workflow

This topic describes the elements and types that make up a variable definition. Variables can be referenced by workflow steps for substitution in step instructions and templates, and for calls to REST interfaces. A workflow can contain up to 1500 variable definitions.

This topic includes the following information:

- "Using Velocity templates for variable substitution and other functions"
- "Specifying the variable element and its attributes" on page 633
- "Sub-elements of the variable element" on page 634
- "Using the element atCreate to qualify a variable definition" on page 636
- "How to refer to a variable" on page 637
- "Providing a workflow variable input file" on page 639

The elements and attributes that are used to define variables are listed in Table 336 on page 675 and Table 337 on page 677.

Using Velocity templates for variable substitution and other functions

The z/OSMF symbol replacement mechanism uses the open source Velocity engine created by the Apache Velocity Project. Though the Velocity engine can be used for simple string replacement, it also contains conditional directives that allow you to generate different strings, based on the presence or value of any variable that is referenced by the step. The type of the variable, as defined in the XML, is passed in to the Velocity engine so that the expected behavior is preserved, except for time and date, which are passed in as strings.

The following example shows a simple step with manual instructions.

```
<step name="Step1" >
        <title>
            Define the started task user ID to SAF.
        </title>
        <description>
            Define the started task user ID to SAF.
        </description>
        <instructions>
            You must define the user ID to your security product.
            For example, for RACF:<br/>
                  ADDUSER STASK OMVS(UID(18136) HOME(/u/stask))
            <br/><br/>
            After you have entered this command from the TSO command line,
            press <strong>Finish</strong> to complete the step.
        </instructions>
        <weight>2</weight>
        <skills>Security administration</skills>
</step>
```

Suppose that you want to prompt the user for input, which might then be substituted in the command image that is contained in the instructions. To do so, you can modify this step to include a variable. In the example that follows, the reference to `UID 18136` in the previous example is replaced with a variable that is used to prompt the user for a UID.

```
<step name="Step1" >
        <title>
            Define the started task user ID to SAF.
        </title>
        <description>
            Define the started task user ID to SAF. You will be
           prompted for the UNIX UID to assign to the user.
        </description>
        <variableValue name="uid" required="true"/>
        <instructions substitution="true">
            You must define the user ID to your security product.
            For example, for RACF:<br/>
                  ADDUSER STASK OMVS(UID($instance-uid) HOME(/u/stask))
            <br/><br/>
            After you have entered this command from the TSO command line,
            press <strong>Finish</strong> to complete the step.
        </instructions>
        <weight>2</weight>
        <skills>Security administration</skills>
</step>
```

In the example, observe the following considerations:

- User input is defined by using the `variableValue` element. In this example, the variable is named `uid`.
- Substitution is performed by using a variable reference, which is `$instance-uid` in this example.

  A variable reference follows this format:
  - Dollar sign ($)
  - Scope, which is either `instance` or `global`
  - Hyphen (-)
  - Variable name, for example, `uid`.

For more examples of how to code symbolic variable references within instructions and templates, see file `workflow_sample_wizards.xml`, which is supplied with z/OSMF in the `/samples` subdirectory of the product file system.

Note:

- When you are using Velocity comparison operators in the instructions, do not use the less than ("<") and greater than (">") characters, as they interfere with XML. Instead, use the alternative notation: `lt`, `le`, `gt`, and `ge`.

- White space (newlines, tabs, spaces) is collapsed before text is displayed in the Workflows task. The indenting that is shown in this example is included for readability only. To obtain the desired spacing for your workflow in the Workflows task, you must provide the appropriate HTML formatting tags. You might need to experiment with the spacing somewhat.

For more information about the Velocity engine, see the following website: `http://velocity.apache.org`.

Specifying the variable element and its attributes

This topic describes the element that defines a variable.

A variable is defined on the variable (`<variable>`) element.

The elements and attributes that are used to define variables are listed in Table 336 on page 675 and Table 337 on page 677.

The following attributes are supported for the variable (`<variable>`) element:

name Name of the variable. The name is required, and must be a string consisting of letters (uppercase or lowercase), numeric digits, the hyphen, and the underscore character. This value must begin with a letter.

The combination of name and scope must be unique within the workflow.

scope Scope of the variable, as follows:

instance
Variable is used only within the workflow in which it is defined. If multiple workflows are created from the same workflow definition file, each has its own set of instance scoped variables.

global Variable can be referenced by any workflow that is imported into the Workflows task. Global variables are shared across all workflows, even workflows that are created from different workflow definitions. As an example, you might use a global variable to refer to a product-specific constant across a number of workflows that are associated with the product.

The scope is required. The default is *instance*.

The combination of name and scope must be unique within the workflow.

You cannot use the same name for both an instance variable and a global variable in the same workflow definition.

Use global variables with caution to avoid possible naming conflicts across unrelated workflows. Consider your naming conventions carefully and avoid using unspecific variable names. Consider qualifying your variables, for example, with the 3-character prefix associated with your software product, or a similarly unique identifier.

Be aware that variables are case-sensitive. For example, "Variable1" is not the same as "variable1."

visibility
Specifies whether the variable is intended for public or private use. This attribute is intended for the workflow author's use. The visibility setting does not affect how the variable is processed by the Workflows task. This attribute is optional; the default is *private*.

Example of a variable definition

In the example in Figure 319 on page 634, the variable `variable_test` is defined.

```
<variable name="variable_test" scope="instance">
        <label>Variable 4</label>
        <abstract>Abstract for Variable 4.</abstract>
        <description>Description for Variable 4.</description>
        <category>variables</category>
        <string/>
    </variable>
```

Figure 319. Specifying attributes on the variable element

Sub-elements of the variable element

This topic describes the sub-elements and types that make up a variable definition.

The `<variable>` element can contain the following sub-elements:

- label (required)
- abstract (required)
- description (required)
- exposeToUser (optional)
- category (required)
- datastore (optional)

Variables require a label (`<label>`) and an abstract (`<abstract>`). These values are displayed in the Workflows task when it prompts for input. In addition, the Workflows task displays a description (`<description>`) for the variable if the user clicks the information icon for the abstract.

You can specify a category (`<category>`) for a variable to assign it to a logical group of related variables. For a given step, all variables with the same category are displayed on the same web page. When viewed through the Workflows task, the workflows Step Perform wizard proceeds through each of the categories that you define for the step. In this way, you can logically organize many variables to provide context and an easier user experience for users who enter variable values.

Variable definition type-specific elements

A variable definition includes a type-specific element that contains elements and attributes specific to that type. The supported types and corresponding element names are described as follows:

`boolean`
: The Workflows task displays a simple check box to prompt the user for this variable. Specify a default value of `true` or `false` to indicate whether the check box is initially displayed to the user as checked. If you do not provide a value, `true` is used by default.

`string`
: The Workflows task displays the variable in a text box, initially primed with an optional default value, if you specify one. Use the "multiLine=" attribute (Boolean) of the string element (`<string>`) to specify whether the text box is small or large.

  You can specify a number of choices for the variable. If so, the Workflows task displays the text box with a menu from which the user can select a value for the variable. The "valueMustBeChoice=" attribute (Boolean) of the string element specifies whether the user must choose from the predefined values or can enter a custom value.

  You can specify more validation criteria for the variable in one or more of the following ways:

  - Minimum length (`<minLength>`) or maximum length (`<maxLength>`), or both, of the string value

- Predefined validation type (<validationType>) to be provided by the Workflows task. You can request validation for common constructs, such as data set names, data set qualifiers, z/OS user IDs. For a list of available validation types, see "Variable definition elements and types summary" on page 674.
- Regular expression (<regularExpression>) that you provide when neither of the other mechanisms meet your requirements. The regular expression must adhere to the JavaScript standard; see the document posted at http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

The criteria that you specify are enforced by the Workflows task in real time as the user types the input value. If the user specifies incorrect syntax, for example, the Workflows task displays the variable in red, along with a default error message, which you can override by providing the error message element (<errorMessage>). Any default or choice value that you specify in the variable definition is also subject to these criteria.

The Workflows task cannot load an XML file that violates the criteria. Similarly, you cannot define the "valueMustBeChoice=" attribute with a value of true without defining any choices.

integer
The Workflows task displays the variable in a text box, initially set to a value that you can optionally specify. The Workflows task restricts the user's input value to a signed 31-bit value (in the range of -2147483648 to 2147483647). You can optionally specify a minimum value (<minValue>) and maximum value (<maxValue>) for the integer. The Workflows task validates the default value against the minimum and maximum when the workflow definition file is imported into z/OSMF.

decimal
The Workflows task displays the variable in a text box, initially set to a value that you can optionally specify. A decimal is an integer with a "decimalPlaces=" attribute on the <decimal> element. A decimal value allows the same whole number value as an integer, plus up to six decimal places (that is, a value in the range of -2147483648.999999 to 2147483647.999999). The "decimalPlaces=" attribute has a default value of 1. You can optionally specify a minimum value (<minValue>) and maximum value (<maxValue>) for the decimal. The Workflows task validates the default value against the minimum and maximum when the workflow definition file is imported into z/OSMF.

time
The Workflows task displays the variable in a timebox. By default, the time is displayed in 15-minute increments, based on your specified default, or the current time, if you do not specify a default. The user can type in a value, also. You can optionally specify a minimum value (<minValue>) and maximum value (<maxValue>) for the time. The Workflows task restricts the user input to the range you specify.

Note:
- You specify this variable in hours-minutes-seconds format (hh:mm:ss), but the timebox displays the time in a slightly different format.
- The schema allows slight variations of this format, but the Workflows task does not. Using a time format other than hh:mm:ss can have an unpredictable result.

date
The Workflows task displays the variable in a calendar, for which the date is based on your specified value, or the current date, if you do not specify a value. The user can type in a value, also. You can optionally specify a minimum value (<minValue>) and maximum value (<maxValue>) for the date. The Workflows task restricts the user input to the range you specify.

Note:
- You specify this variable in year-month-day format (yyyy-mm-dd), but the calendar displays the date in a slightly different format.
- The schema allows slight variations of this format, but the Workflows task does not. Using a date format other than yyyy-mm-dd might have an unpredictable result.

password
By defining a password variable, you can add a password prompt to your workflow. If you do so, the user is prompted to provide a password on the Input Variables tab of the Workflows task. In the user interface, the password variable is displayed as an input field. For a Provisioning workflow, the input field replaces the user's typed characters with masking characters, such as asterisks ('********'). For General workflows and Configuration workflows, the input field is not masked; the password is shown as it is typed.

You must specify either of the following types of validation checking for a password variable:

- Minimum length (<minLength>) or maximum length (<maxLength>), or both, of the password value.
- Match with a regular expression (<regularExpression>). The expression must adhere to the JavaScript standard; see the document posted at http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

The criteria that you specify are enforced by the Workflows task in real time as the user types the input value. If the user specifies an incorrect syntax, for example, the Workflows task displays the variable in red, along with a default error message, which you can override by providing the error message element (<errorMessage>).

The password variable type has no default value.

Using the element atCreate to qualify a variable definition

This topic describes the element atCreate (<atCreate>). For users of the Create Workflow REST service, the atCreate element provides additional options for working with variables.

The Create Workflow REST service is described in "Create a workflow" on page 528.

The following attributes can be specified on the atCreate element:

name Specifies the variable for which the variable attributes are being set. The name is required. For example, to set a variable named var1, define the atCreate element with the name var1.

scope Specifies the scope of the variable. This value is set to *instance* (the only valid value) or is omitted; the default is *instance*.

required
This attribute has a specialized purpose. For a workflow that is created through the Create Workflow REST service, this attribute indicates whether the variable must be set to a value at the time of workflow creation. A variable can be set in any of the following ways:

- Defining a default value for the variable in the workflow definition
- Setting a value for the variable in the workflow variable input file
- Specifying a value for the variable on a Create Workflow service.

This attribute is optional; the default is *false*. If a variable is marked as "required," but the variable is not given a value, an attempt to create the workflow through the Create Workflow REST service will fail with an error.

For a workflow that is created through the Workflows task user interface, this option is ignored. That is, the workflow is created, regardless of the setting of the required attribute.

Note: This setting is returned as the "requiredAtCreate" property of a variable by the Retrieve Workflow Definition service; See "Retrieve a workflow definition" on page 560.

prompt
For users of the Create Workflow REST service, this attribute identifies a variable that *should* be prompted for by the program that issues the REST service. By itself, the prompt attribute does not enforce any behavior for the workflow creation. However, by setting prompt to *true*, you can indicate that prompting is recommended for the variable. The user of the Create Workflow REST

service can query the value of the prompt attribute for any variables in the workflow to determine whether any variables should be prompted for.

This attribute is optional; the default is *false*.

Note: This setting is returned as the "promptAtCreate" property of a variable by the Retrieve Workflow Definition service; See “Retrieve a workflow definition” on page 560.

You can specify the atCreate element for any instance variables that are used in a workflow definition. The atCreate element is not valid for global variables.

Example of using the element atCreate

Suppose that you have a variable that would be useful to include in a number of different workflows. If so, you can define the variable in an XML file (an XML fragment) and include the same fragment in the appropriate workflow definitions. The variable is now shared between these workflows.

In Figure 320, the variable `variable_test` is defined in an XML file. The variable is used by more than one workflow, so the variable definition is coded in an XML file that can be included with multiple workflows.

```
<variable name="variable_test" scope="instance">
        <label>Variable 4</label>
        <abstract>Abstract for variable 4.</abstract>
        <description>Description for variable 4.</description>
        <category>variables</category>
        <string/>
    </variable>
```

Figure 320. Variable definition in this example

Now suppose that the variable's attributes for *required* and *prompt* need to be set differently for different workflows. The atCreate (`<atCreate>`) element, which is used with the variable element, allows you to specify different *prompt* and *required* settings for the same variable in different workflow definitions. To do so, have each workflow definition include the XML file that defines the variable. Then, in each workflow definition, specify the atCreate element to further clarify the properties of the variable.

In Figure 321, the atCreate element is used to specify the attributes `required` and `prompt` for the variable `variable_test`, which was defined in Figure 320. The atCreate element can be defined differently in any workflow definition that refers to the variable `variable_test`.

```
<atCreate name="variable_test" required="true" prompt="true"/>
```

Figure 321. How the atCreate element is used to specify variable attributes for required and prompt

By including the atCreate element in each workflow definition, you can set different values for the *prompt* and *required* attributes for the same variable in different workflows.

How to refer to a variable

For examples of the various variable types and features, see file `workflow_sample_variables.xml`, which is supplied with z/OSMF in the `/samples` subdirectory of the product file system.

Using braces around variable references is optional, but recommended as a good programming practice. The braces help to ensure that variables are clearly identified in the workflow. More importantly, the braces prevent ambiguity when it comes time for variable substitution, such as in conditional expressions, and jobs and scripts. For example, the variables `$st_userFRED` and `${st_user}FRED` are evaluated

differently by the Workflows task. In the former case, the Workflows task searches for a variable called st_userFRED. In the latter case, it is clear that the variable is st_user.

A variable reference can contain an underscore or a hyphen. For example, both of the following references are valid: ${instance_st_user} == "IBMUSER" and ${instance-st_user} == "IBMUSER".

More examples of how variables are referenced and used are provided in "Defining steps for your workflow" on page 608.

Simplified instance variable format in substitution and conditions

If you need to define many instance variables in a workflow definition file, you can save some typing by using the simplified variable format. That is, you can omit the prefix instance- from the names of instance variables. To use this variable name format, you must enable it by including the optional schema element (<workflowSettingInfo>) with its only subelement (<variablesSetting>) and attribute isInstanceWithoutPrefix set to "true". You must also ensure that none of the instance variables in the workflow definition are prefixed by instance-, either in variable definitions or in conditional expressions.

Note: Though the simplified variable format is supported, it is recommended that you use the standard format, for example: "${instance-varName}" or "${global-varName}", as a good programming practice.

Use care with the simplified variable format when specifying variables in substitutions, to ensure that the variables are specified consistently. In the following example, the simplified format is used for instance variable references. In the example, Step 3 is not performed unless Step 1 and Step 2 complete with a return code zero and the instance variable ${st_user} is IBMUSER.

```
<Step name=Step3">
 <title>A conditional step based on return code</title>
 <description>This conditional step is not ready unless
         the two preceding steps complete with RC 0
               variable st_user value is IBMUSER
 </description>
 <instructions>Run this job.</instructions>
 <condition>
   <expression><![CDATA[${step1.returnCode} == "0000" &&
                  ${step2.returnCode} == "0000" &&
                  ${st_user} == "IBMUSER" ]]>
     </expression>
  <description>This conditional step is not ready unless the two
                preceding steps complete with RC 0 and the variable
                st_user value is IBMUSER.</description>
 </condition>
 <targetState>Ready</targetState>

⋮
<template>
<inlineTemplate substitution="true">
//STEP3 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=A
//SYSTSIN DD *
ADDGROUP ${st_group} OMVS(GID($st_gid))
/*
</inlineTemplate>
<submitAs>JCL</submitAs>
</template>
</step>
```

Figure 322. You can use variable values in the condition to be satisfied.

Providing a workflow variable input file

This topic describes the format of the editable properties file that is called the *workflow variable input file*. With this file, you can supply users with preset values for the variables that are defined in your workflow definition file. By including a variable input file with your workflow definition file, you save users from having to manually enter values for some or all of the variables in your workflow.

How a workflow variable input file is used

A workflow variable input file is an optional properties file that you, the workflow author, can use to pre-specify one or more of the input variables that are defined in the workflow definition. By supplying variable values in this way, you allow the user to create a workflow without having to interactively enter the inputs in the Workflows task Step Perform wizard. The Workflows task treats any variables set through the workflow variable input file the same as if the user entered them manually.

If you provide a workflow variable input file, include it with the other materials that you supply to the workflow user, such as the workflow definition file and the other files that comprise your workflow definition. Ensure that the documentation for your workflow definition makes note of the file name, and provides any related instructions for editing or storing the file on the user's z/OS system. The Workflows task accesses the workflow variable input file under the user's identity, thus, the file must be read-accessible by the user who is creating the workflow.

At workflow creation time, the user imports the workflow variable input file into the Workflows task, along with the workflow definition file. The Workflows task reads in the contents of the file and saves its values for use with the created workflow. The Workflows task uses the variable input file in addition to any global variables that are already defined to Workflows task. Any new variables that are defined with a global scope become available to the other workflow instances on the user's system. After the Workflows task imports the file, the task no longer refers to the file.

Creating a workflow variable input file

As the workflow author, you can create a workflow variable input file as a text file, using an editor of your choice. The file must be encoded in either of the following formats: ASCII or IBM-1047 (EBCDIC). Use a file type of `.txt` or `.properties`. Do not use Unicode encoding, such as UTF-8, for this file.

In the variable input file, specify the properties (variables and their respective values) as one or more key-value pairs. Valid separator characters are equal signs (=), colons (:) or blanks. Figure 323 shows the valid formats for specifying properties in the variable input file.

```
key1 = value1
key2 : value2
key3   value3
```

Figure 323. Format of a workflow variable input file

Figure 324 shows an example of the contents of a workflow variable input file.

```
Boolean1 = false
String3  = SYS1.LINKLIB
Integer2   35
Decimal2 : 3.3
Time2      03:03:00
Date1 = 2013-11-11
```

Figure 324. Example of a workflow variable input file

The example in Figure 324 on page 639 is designed to work with the file `workflow_sample_variables.xml`, which is supplied as a sample with z/OSMF. For a description of this file and other coding examples, see "Sample XML files for your reference" on page 597.

The Workflows task does no syntax checking of the properties that are specified in the workflow variable input file. Therefore, you must ensure that valid values are specified for each of the properties.

Also, observe the following considerations:

- Each property that is specified in the variable input file must correspond to a variable named in the workflow definition file. Otherwise, the Workflows task ignores the property.
- If the variable input file specifies a property that matches a variable that is already defined to the Workflows task as a global variable, the Workflows task detects the conflicting definitions and prompts the user for a selection. See "Avoid conflicting variable definitions" on page 641

You can provide the workflow variable input file in either a z/OS UNIX file or a z/OS data set. For a z/OS data set, use a sequential data set, a member of a partitioned data set (PDS), or partitioned data set extended (PDSE).

If you create the workflow variable input file on a workstation, it is recommended that you use File Transfer Protocol (FTP) in binary mode to transfer the XML files to a z/OS system. Doing so helps to ensure that the files are encoded properly for use on z/OS.

Saving properties to an output file

As an alternative to defining properties in a workflow definition file or a variable input file, you can design steps to define properties dynamically within a workflow. That is, you can add a step to run a job to create properties and settings, and save the properties in an output file. Doing so makes the properties available for use by other steps in the same workflow instance (as instance variables) or other workflows (as global variables).

To specify a default data set name or UNIX path name for the output file, include the optional subelement `<output>` on the `<step>` element.

When an output file is saved, the Workflows task scans the file to determine whether the file contains any workflow input variables (properties that are written as name-value pairs). If so, the Workflows task attempts to add the variables to the variables pool. The input variables then become available for use by subsequent steps in the workflow instance. In practice, a step might submit a batch job to create some input variables, which are then used by a subsequent step, thus saving the Workflows task user from having to enter the input variables manually.

Create step-specific values with the _output variable

When you include the `<output>` element in the step definition, a special use variable called *_output* is defined implicitly. The _output variable takes on the value specified on the `<output>` element.

For a step that saves its properties to an output file, you can use the `<output>` element to name the output file created by the step. The `output` element specifies a default data set name or UNIX path name for the output file, as described in "Saving properties to an output file ." The end user can override the value of the `<output>` element and choose a different output file, if desired.

For example:

```
<output>some.file.${instance-var1}</output>
```

In this example, if the instance variable var1 has the value "name," the variable _output has the value "some.file.name" after substitution. You can use `${_output}` within the step, perhaps within the body of a

script. Note that the variable has a step specific value. If the element <output> is not specified for the step, the variable _output is not available for that step.

You can find an example of this technique in the **workflow_sample_output.xml** file that is provided with z/OSMF. The following figure shows a portion of the sample XML file:

```
<!-- Declare a set of variables for use -->
<variable name="outputFileNameVariable">
    <label>File name for output file</label>
    <abstract>Enter a name for the output properties file</abstract>
    <description>This value is required. It is included in the JCL for output file path.
    </description>
    <category>Output File</category>
    <string>
        <default>testOutput</default>
    </string>
</variable>
⋮
<variable name="st_user">
⋮
<variable name="setting0" scope="global">
⋮
<step name="submitEmptyJCL">

⋮
<!-- This step demonstrates how to load variables from a JCL/REXX/Shell script execution".
<step name="output">
    <title>A step with an output property file</title>
    <description>In this step, you run a job to create an output property file
    </description>
    <variableValue name="st_user" required="true"/>
    <variableValue name="setting0" required="true" scope="global"/>
    <variableValue name="outputFileNameVariable" required="true"/>
    <instructions>In this example step, some variable values are generated into a pre-specified
                  output file and after perform, those variables will be loaded into workflows
                  task after user resolves the variable conflicts if any.</instructions>
    <weight>5</weight>
    <skills>System Programmer</skills>
    <template>
            <inlineTemplate substitution="true">
# shell script to generate some variables from workflows task to the output property file
echo st_user USER001 >> ${_output}
echo setting0 global001 >> ${_output}
            </inlineTemplate>
            <submitAs maxRc="0">shell-JCL</submitAs>
            <output substitution="true" needResolveConflicts="false">
                        /u/tmp/${instance-outputFileNameVariable}
            </output>
            </template>
</step>
```

As shown in the example, you can use the _output variable in the shell script to assist with producing the properties file. The script will use the output file name as specified by the user who performed the step.

Avoid conflicting variable definitions

If the Workflows task detects that an imported variable conflicts with an existing global variable, the user is prompted to choose the appropriate value. The user can select to use the input file variable in place of existing global variables, or ignore the input file variable, and use the existing global variable instead. The user's selection determines which version of the variable is saved in the Workflows task global variable pool for use with other workflow instances. Thus, the user's selection affects any other workflows that refer to the same global variable.

It is recommended that you choose unique names for variables to avoid possible naming conflicts with unrelated workflows. Consider your naming conventions carefully and avoid using unspecific variable names. Consider qualifying your variables, for example, with the three-character prefix associated with your software product, or a similarly unique identifier.

Depending on your design, you might determine that the output file variables must always be used in place of a workflow's existing instance variables. If so, you can include the need-resolve-conflicts attribute (`needResolveConflicts`) on the `output` subelement, and set it to false. If so, the Workflows task uses the output file variables in place of any existing values without prompting the user. This setting applies to instance variables only; global variables are not overridden. The default is true; if variable conflicts exist, the user is prompted to resolve the conflicts.

Workflow XML reference

This language reference describes the elements and attributes that comprise a workflow definition.

This topic is organized in tables, with each table describing a major portion of the Workflows schema—the elements and their attributes, default values, the XML attribute data types, and whether a particular attribute is required. Table 325 lists the reference tables.

Table 325. Reference tables for the Workflows XML schema

Element type	Description	Where described
Workflow metadata elements	The elements that make up the workflow metadata	Table 326 on page 645
Sub-elements for a configuration type workflow	For a configuration type workflow, these sub-elements are required	Table 327 on page 648
Workflow upgrade elements	The elements that define the workflow upgrade options	Table 329 on page 651
Manifest elements	The elements that make up the manifest	Table 330 on page 653
Step elements: Elements to use for defining all steps	The elements to use for defining all steps	Table 331 on page 655
Step elements: Additional sub-element for parent steps	The step sub-element, which is used to identify the containing step as a parent step	Table 332 on page 656
Step elements: Additional sub-elements for leaf steps	The sub-elements to use for defining a leaf step, which is a step that does not contain step elements	Table 333 on page 657
Step elements: Additional sub-elements for REST steps	The sub-elements to use for defining a step that issues a REST request, such as GET or PUT. This type of step is referred to as a *REST step*.	Table 334 on page 668
Step elements: Additional sub-elements for steps that invoke another workflow	The sub-elements to use for defining a step that invokes another workflow, which is referred to as the *called workflow*	Table 335 on page 671
Variable definition elements	The elements that make up a variable definition	Table 336 on page 675
Variable definition type-specific sub-elements	The type-specific sub-elements that make up a variable definition	Table 337 on page 677
atCreate element	For users of the Create Workflow REST service, the atCreate element provides additional options for working with variables.	Table 338 on page 682

In the tables that follow, the elements are listed in the order in which they are required by the schema. Though you can omit optional elements, the elements that you specify must follow the order in which the elements are presented. In contrast, the attributes within an element can be specified in any order.

Note: The tables in this language reference are formatted in landscape view to improve usability when you print copies of these pages. To adjust the view in Adobe Reader, select **View** > **Rotate View** > **Clockwise**.

Workflow metadata elements summary

Table 326. Workflow metadata elements

Element name	Description	Required or optional	Type	Supported attributes
workflowInfo	Contains the workflow metadata.	Required.	Contains the following sub-elements: • workflowID (required) • workflowDescription (required) • workflowVersion (required) • vendor (required) • configuration or general (optional)	No attributes are supported for this element.
workflowID	A short, arbitrary value that identifies the workflow.	Required.	nonNullString	The following attributes are supported for the *workflowID* element: **scope** Indicates the singleton scope for the workflow. The following values are valid: **system** A maximum of one instance of this workflow can exist on any one system in the sysplex. **sysplex** A maximum of one instance of this workflow can exist in the sysplex. **none** No limit exists for the number of instances of this workflow. For a callable workflow, this setting means that a new instance is always created on the calling system. The default setting is *none*. Omitting the scope attribute has the same effect as the default setting. **isCallable** Indicates whether the workflow can be called by another workflow, and, if so, the callable range for the workflow. The following values are valid: **system** This workflow can be called only by another workflow that is running on the same system. **sysplex** This workflow can be called by any workflow that is running in the same sysplex.
workflowDescription	A short description of the workflow.	Required.	nlsString	No attributes are supported for this element.

Table 326. Workflow metadata elements (continued)

Element name	Description	Required or optional	Type	Supported attributes
workflowVersion	The version of this workflow definition file. Update this value whenever you change any portion of the workflow definition file, including changes to the primary XML file or any sub-files or referenced files. The Workflows task caches only the latest version of any imported workflow definition file. Therefore, to ensure that the most current version is used, you must update the version value whenever you modify the workflow definition. For this reason, when you author a workflow definition file, you might want to complete the development phase on a workstation before you import the workflow definition into the Workflows task.	Required.	nonNullString	No attributes are supported for this element.
vendor	The name of the workflow provider.	Required.	nonNullString	No attributes are supported for this element.
Workflow category is a classification of the activities to be performed in the workflow. Specifying a workflow category element is optional. By default, the workflow category is *general*. To indicate a category, specify one of the following elements: **configuration** A workflow that is used to configure system software is classified as a *configuration workflow*. **provisioning** A workflow that is used to provision system software is classified as a *provisioning workflow*. **general** All other workflows are classified as *general workflows*. The category elements are described in the next rows.				
Configuration	A workflow that is used to configure system software is classified as a *configuration workflow*.	Optional.	Contains the sub-elements listed in Table 327 on page 648.	No attributes are supported for this element.

Table 326. Workflow metadata elements (continued)

Element name	Description	Required or optional	Type	Supported attributes
Provisioning	A workflow that is used to provision system software is classified as a *provisioning workflow*.	Optional.	Contains the sub-elements listed in Table 328 on page 649.	No attributes are supported for this element.
General	All other workflows are classified as *general workflows*.	Optional.	Empty	No attributes are supported for this element.

Table 327. Sub-elements for a configuration type workflow

Element name	Description	Required or optional	Type
productID	A short, arbitrary value that identifies the workflow.	Required.	nonNullString
productName	The name of the product	Required.	nonNullString
productVersion	The product version	Required.	nonNullString

Table 328. Sub-elements for a provisioning type workflow

Element name	Description	Required or optional	Type
productID	Identifier of the product or component that is being provisioned by the workflow, such as the product identifier (PID) or function modification identifier (FMID).	Required.	nonNullString
productName	Name of the product or component that is being provisioned by the workflow.	Required.	nonNullString
productVersion	Version and release of the product or component that is provisioned by the workflow.	Required.	nonNullString
softwareType	Type of software to be provisioned by the workflow.	Required.	

Workflow upgrade elements summary

Table 329. Workflow upgrade elements

Element name	Description	Required or optional	Type
preserveOptions	Contains the workflow upgrade options.	Optional.	Contains the following sub-elements: • version (required) • variableSet (optional) • stepSet (optional) • workflowHistory (optional) • workflowNotes (optional) • include (required) • exclude (optional) • upgradeNotes (optional)
version	Identifies the workflow version that can be upgraded by this workflow definition file.	Required; at least one.	Contains the following sub-elements: • value (required) • type (required)
variableSet	The variables to copy to new workflow instance.	Optional.	Contains the following sub-element: • defaultChecked (optional)
stepSet	The steps to copy to new workflow instance.	Optional.	Contains the following sub-element: • defaultChecked (optional)
workflowHistory	Specifies whether to copy the workflow history from the existing workflow to the new instance.	Optional.	Contains the following sub-element: • defaultChecked (optional)
workflowNotes	Specifies whether to copy the workflow notes from the existing workflow to the new instance.	Optional.	Contains the following sub-element: • defaultChecked (optional)
include	Specifies the step or variable defined by prior workflow definition file to be copied. Can be specified multiple times. It supports regular expression or variable name.	Required.	Contains the following sub-elements: • name (optional) • mapTo (optional) • regExp (optional)
exclude	Specifies the variables to exclude from the set generated by <include> elements.	Optional.	Contains the following sub-element: • name (required)

Manifest elements summary

Table 330. Manifest elements summary table

Element name	Description	Required or optional	Type	Supported attributes
translatedTextFiles	Contains the language file definitions.	Optional.	If specified, it must contain the following sub-elements: • bundle (required) • language (required)	No attributes are supported for this element.
bundle	Contains the set of language files for the bundle	Required if the translatedTextFiles element is specified. A workflow can contain 1 – 500 bundles.	A sequence of language elements.	The following attribute is supported for the *bundle* sub-element: **name** The name of the bundle. The name is required, and must be a single-token string.
language	Locates a file for a particular language.	Required if the bundle element is specified. A bundle can contain 1 – 10 languages.	Empty	The following attributes are supported for the *language* sub-element: **name** The language identifier as defined in the XML standard and RFC 3066, but using only the language portion without the country suffix. The identifier is required, and must be unique within a bundle. **path** The path name of the language file. The path is required, and its data type must be a nonNullString. A language file can be a UNIX file, a sequential data set, or a PDS member. The path name format is described in "References to external files" on page 598.

Step elements summary

Table 331. Step elements summary table: Elements to use for defining all steps

Element name	Description	Required or optional	Type	Supported attributes
step	Contains attributes of a step. Up to 500 step elements can be defined in a workflow.	Required. At least one step must be defined.	The step element can contain the following sub-elements: • title (required) • description (required) • prereqStep (optional) • If a step is a parent step (contains step elements), use the sub-element listed in Table 332 on page 656 to define the step. • If a step is a leaf step (contains no step elements), use one or more of the sub-elements listed in Table 333 on page 657 to define the step. • If a step invokes another workflow, use one or more of the sub-elements listed in Table 335 on page 671 to define the step.	The following attributes are supported for the *step* element: **name** The name of the step. The name is required, and must be a string consisting of letters (uppercase or lowercase), numeric digits, the hyphen, and the underscore character. This value must begin with a letter, and must be unique within the workflow. **optional** Indicates whether the step is optional. This attribute is optional. If specified, its data type must be Boolean. By default, the value is *false*.
title	A short description of the task.	Required.	nlsString	No attributes are supported for this sub-element.
description	A more detailed description of the step.	Required.	nlsRichString	No attributes are supported for this sub-element.
prereqStep	Identifies a step that must be completed before this step can be performed. Up to 499 prerequisite steps can be defined for a step.	Optional.	Empty	The following attribute is supported for the *prereqStep* sub-element: **name** Name of the prerequisite step. The name is required, must refer to a defined step, and must be a string consisting of letters (uppercase or lowercase), numeric digits, the hyphen, and the underscore character. This value must start with a letter.

Table 332. Step elements summary table: Additional sub-element for parent steps

Element name	Description	Required or optional	Type	Supported attributes
step	A substep. Existence of this sub-element defines a parent step. This sub-element is subject to the 500 step maximum. The step sub-element is mutually exclusive with template, instructions, variableValue, skills, and weight.	Required.	Recursive step definition as defined by this table.	No attributes are supported for this sub-element.

Table 333. Step elements summary table: Additional sub-elements for leaf steps

Element name	Description	Required or optional	Type	Supported attributes
runAsUser	Specifies the user ID under which the step is to be performed. This value is used for automated steps; it allows for a step to be performed by a user other than the step owner. When not specified, the default is the step owner user ID.	Optional.	runAsUserType	None.
condition	A conditional step.	Optional.	Condition for performing this step.	The following attributes are supported for the *condition* sub-element: **expression** Provides an expression based on input variables (global or instance) and Boolean logic. When the expression resolves to *true* for the current workflow instance, the step can be performed. An expression is required, and must have the data type conditionType. **description** Describes the condition that must be satisfied before the step can be performed. A description is required. **targetState** The state to which the step is set when the expression evaluates to *true*. The valid values are *Ready* and *Skipped*. The target state is optional. If not specified, the default is *Ready* .

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
variableValue	References a variable that is defined earlier. The variableValue sub-element is mutually exclusive with the step sub-element.	Optional.	Empty	The following attributes are supported for the *variableValue* sub-element: **name** The name of the referenced variable. The name is required, and must be a string consisting of letters (uppercase or lowercase), numeric digits, and the underscore character. This value must begin with a letter. The name and scope combination must refer to a defined variable. **scope** The scope of the referenced variable. The scope is optional. If specified, the value must be *instance* or *global*. The default is *instance*. The name and scope combination must refer to a defined variable. **required** Whether the variable must have a value for this step. The required attribute is an optional, Boolean value. If not specified, the default is *false*. When set to *true*, the Workflows task does not allow the user to complete the step without providing a value (if a default is not defined in the XML). **noPromptIfSet** Whether the variable widget is displayed in read-only mode, if the variable already has a value. The noPromptIfSet attribute is an optional, Boolean value. If not specified, the default is *false*, that is, always display the variable widget in read/write mode.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
instructions	Detailed documentation on what the user must do to perform the step. This sub-element is mutually exclusive with the step sub-element.	Required.	nlsRichVelocityString	The following attribute is provided for the *instructions* sub-element: **substitution** Indicates whether instructions contain variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false.* A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.
weight	The relative difficulty of the step compared to other steps within this workflow. The weight sub-element is mutually exclusive with step sub-element.	Required.	Integer value 1 - 1000.	No attributes are supported for this sub-element.
skills	The type of skills that are required to perform this step. The skills sub-element is mutually exclusive with step sub-element.	Optional.	nlsString	No attributes are supported for this sub-element.
autoEnable	Indicates whether the step is to be performed automatically when all prerequisite steps are completed, and no user inputs are required. If autoEnable is not specified, the default is *false*.	Optional.	Boolean	No attributes are supported for this sub-element.
canMarkAsFailed	Indicates whether the step can be marked as *Failed* manually by the step owner. If canMarkAsFailed is not specified, the default is *false.* When set to *true*, the Review Instructions page in the Step Perform wizard includes the option to allow the step owner to mark the step as *Failed* manually. When *false*, this option is not displayed to the user.	Optional.	Boolean	No attributes are supported for this sub-element.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
rest	Identifies the step as a REST step.	Optional.	String.	The sub-elements and attributes of a REST step are described in Table 334 on page 668.
template	Identifies the step as a *template step*, which is a step that runs an executable program, such as a JCL job, a REXX exec, or a UNIX shell script. The template sub-element is mutually exclusive with the step sub-element.	Optional.	The template sub-element must contain one of the following sub-elements: **inlineTemplate** A file template or executable template that is specified in the workflow definition file. The value must be type velocityString. **fileTemplate** Path name of the external file that contains the template. The contents of the file are treated as type velocityString.	The following attribute is supported for both the inlineTemplate and fileTemplate sub-elements: **substitution** Indicates whether the template contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
submitAs	Indicates the type of executable program.	Optional.	**"TSO-REXX"** Run a REXX exec program in real time. **"TSO-UNIX-REXX"** Run a REXX exec program for the UNIX environment in real time. **"TSO-UNIX-shell"** Run a UNIX shell script in real time. **"JCL"** Submit a JCL job for batch processing on z/OS. The results are indicated in the job log. **"TSO-REXX-JCL"** Submit a JCL job that contains a REXX program. The program runs as a batch job on z/OS; the results are indicated in the job log. **"shell-JCL"** Submit a JCL job that contains a UNIX shell script. The program runs as a batch job on z/OS; the results are indicated in the job log. A REXX exec that is written to be run in a UNIX shell environment should be submitted as "TSO-UNIX-shell" or "shell-JCL".	The following attribute is supported for the *template* element when the *submitAs* sub-element is "JCL", "TSO-REXX-JCL", or "shell-JCL": **maxRC** Maximum return code value to consider successful. This attribute is optional. If specified, the value must be an integer in the range 0 - 4095. If not specified, the default is 0.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
maxLrecl	For a step that submits a job, this value specifies the maximum record length, in bytes, for the input data for the job. This value is used when the step is performed automatically (*autoEnable=true*). If the step is performed manually, the user can optionally specify the maximum record length on the Edit JCL page in the Workflows task.	Optional.	Integer value 80 - 1024. The default is 1024 bytes.	No attributes are supported for this sub-element.
output	Indicates the default name of the output property file that is produced by the step (a data set or UNIX file). The output file can contain variables and values that are used by subsequent steps. The Workflows task allows the user to modify the file name and location, as needed.	Optional.	String.	The following attributes are supported for the *output* sub-element: **substitution** Indicates whether the output file name contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *true*. **needResolveConflicts** Indicates whether the user is prompted to resolve variable conflicts from the output file. This attribute is an optional, Boolean value. If not specified, the default is *true*. If variable conflicts exist, the user is prompted to resolve the conflicts. A value of *true* must be specified for the Workflows task to display the variables in the Input Variables page. If set to *false*, the Workflows task uses the output file variables in place of any existing values without prompting the user. This setting applies to instance variables only; global variables are not overridden by variables in the output file.
successPattern	Regular expression that is returned for a successful program execution. This element is required. You must specify one (and only one) regular expression for a successful program completion.	Required.	String.	No attributes are supported for this sub-element.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
failedPattern	Optional regular expression that can be returned for program execution failures. You can omit this element or specify up to 100 different specifications for *failedPattern*. This property might be null.	Optional.	String.	No attributes are supported for this sub-element.
outputVariablesPrefix	For a step that creates a variable, this property contains a prefix that identifies a string as a variable. This property might be null.	Optional.	String.	The following attribute is supported for the *outputVariablesPrefix* sub-element: **needResolveConflicts** Indicates whether the user is prompted to resolve variable conflicts from the program output variables. This attribute is an optional, Boolean value. If not specified, the default is *true*. If variable conflicts exist, the user is prompted to resolve the conflicts. A value of *true* must be specified for the Workflows task to display the variables in the Input Variables page. If set to *false*, the Workflows task uses the output file variables in place of any existing values without prompting the user. This setting applies to instance variables only; global variables are not overridden by variables in the output file.
scriptParameters	For a step that runs a program, this property contains the input parameters that can be set by the step owner. This property might be null. The following sub-elements are included on the *scriptParameters* element: • *description* • *value*	Optional.	String.	No attributes are supported for this sub-element.
description	Text description of the parameter in the *scriptParameters* element, such as its intended use or recommended value.	Required if the *scriptParameters* element is specified.	String.	No attributes are supported for this sub-element.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
value	Value of the parameter in the *scriptParameters* element.	Required if the *scriptParameters* element is specified.	String.	No attributes are supported for this sub-element.
procName	For a step that runs a program under TSO/E, this property contains the name of the logon procedure that is used to log into the TSO/E address space. If no value was specified for the step, the default is IZUFPROC.	Optional.	String.	The following attribute is supported for the *procName* sub-element: **substitution** Indicates whether the procedure name contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*.
regionSize	For a step that runs a program under TSO/E, this property contains the region size for the TSO/E address space. If no value is specified for the step, the default is 50000.	Optional.	Integer value 50000 - 2096128.	No attributes are supported for this sub-element.
timeout	For a step that runs a REXX exec or UNIX shell script, this property contains the maximum amount of time that the program can run before it is ended by a timeout condition.	Optional.	Integer value 60 - 3600.	No attributes are supported for this sub-element.
saveAsDataset	Data set name (fully qualified, no quotations) specifying where to save the file after the user edits it. When a file is generated, the presence of this element results in the *save as data set* option being presented to the user, primed with the element value, if specified. When a program is run, this element can be used as the default value. However, the Workflows task widget is always displayed to the user.	Optional.	velocityFileString	The following attribute is supported for the *saveAsDataset* sub-element: **substitution** Indicates whether the data set name contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
saveAsUnixFile	Path name that specifies where to save the file after the user edits it. When a file is generated, the presence of this element results in the *save as UNIX file* option being presented to the user, primed with the element value, if specified. When a program is run, this element can be used as the default value. However the Workflows task widget is always displayed to the user.	Optional.	velocityFileString	The following attribute is supported for the *saveAsUnixFile* sub-element: **substitution** Indicates whether the path name contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.
predefinedVariable	For a step that submits a job, this sub-element sets a variable in the body of the job. A predefined variable is treated as a string substitution. The substitution applies to the current step only. You can specify multiple predefined variables per step.	Optional.	predefinedVariableType	The following attribute is supported for the *predefinedVariable* sub-element: **name** Name of the predefined variable. The name is required, and must be of data type string. To avoid overriding the variables defined for the workflow, use a unique name for the predefined variable.
feedbackItem	Optionally includes a feedback form for the step in the Workflows task with questions for the step owner to answer. Up to 100 feedback items (questions) can be specified for a step.	Optional.	feedbackItemType	The following attributes are supported for the *feedbackItem* sub-element: **name** Name of the feedback question. The name is required, and must be of data type string. **required** Indicates a required question. This attribute is an optional, Boolean value. If not specified, the default is *false*.
question	The question to be asked (a text string). For example: *How difficult was this step?*	Required, if the element *feedbackItem* is specified.	String.	No attributes are supported for this sub-element.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
answers	Answer format, which is defined by including one of the following sub-elements: **singleSelect** One answer (and only one) can be selected from the available choices. **multipleSelect** More than one answer can be selected from the available choices. **text** No choices are provided; the user answers the question by entering a text value.	Required, if the element *feedbackItem* is specified.	answersType	No attributes are supported for this sub-element.
singleSelect	A sub-element of the element *answers*. Indicates that one answer (and only one) can be selected from the available choices listed in *answers*.	Optional.	selectType	The following attribute is supported for the *multipleSelect* sub-element: **hasOtherAnswer** Displays the choice *"Other"* as a selectable answer. Is displayed last in a list of multiple choices.
label	One or more answers to be listed as selectable choices for the question. At least one label (answer) must be specified, up to a maximum of 50.	Required.	labelType	The following attributes are supported for the *label* sub-element: **value** Answer that can be selected. Required. **required** Indicates a required question. This attribute is an optional, Boolean value. If not specified, the default is *false*.
multipleSelect	A sub-element of the element *answers*. Indicates that more than one answer can be selected from the available choices.	Optional.	selectType	The following attribute is supported for the *multipleSelect* sub-element: **hasOtherAnswer** Displays the choice *"Other"* as a selectable answer. Is displayed last in a list of multiple choices.
label	One or more answers to be listed as selectable choices for the question. At least one label (answer) must be specified, up to a maximum of 50.	Required.	labelType	The following attributes are supported for the *label* sub-element: **value** Answer that can be selected. Required. **required** Indicates a required question. This attribute is an optional, Boolean value. If not specified, the default is *false*.

Table 333. Step elements summary table: Additional sub-elements for leaf steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
text	A sub-element of the element *answers*. Indicates that no choices are provided; the user answers the question by entering a text value.	Optional.	String.	No attributes are supported for this sub-element.

Table 334. Step elements summary table: Additional sub-elements for REST steps

Element name	Description	Required or optional	Type	Supported attributes
httpMethod	Indicates the HTTP method that is used for issuing the REST request. The following values are valid: • GET • PUT • POST • DELETE.	Required.	String	None.
schemeName	Scheme name that is associated with the REST request. If specified, this element must be set to "http."	Optional.	String	None.
hostname	Host name or IP address of the system to which the REST request is directed. For example: `www.ibm.com`.	Optional	String	The following attribute is supported for the hostname sub-element: **substitution** Indicates whether the host name contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.
port	Port number to use for the REST request.	Optional.	String	The following attribute is supported for the post sub-element: **substitution** Indicates whether the port contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.

Table 334. Step elements summary table: Additional sub-elements for REST steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
uriPath	URI path to use for the REST request.	Required.	String	The following attribute is supported for the uriPath sub-element: **substitution** Indicates whether the URI path contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.
queryParameters	For a GET or POST request, this element contains the query parameters.	Optional.	String	The following attribute is supported for the queryParameters sub-element: **substitution** Indicates whether the query parameters contain variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.
requestBody	For a PUT or POST request, this element contains the request body.	Optional.	String	The following attribute is supported for the requestBody sub-element: **substitution** Indicates whether the request body contains variable substitution. This attribute is an optional, Boolean value. If not specified, the default is *false*. A value of *true* must be specified for the Workflows task to allow the variable substitution. If *true* is specified, at least one *variableValue* sub-element must be specified for the step.

Table 334. Step elements summary table: Additional sub-elements for REST steps (continued)

Element name	Description	Required or optional	Type	Supported attributes
expectedStatusCode	The expected HTTP status code from the REST API request. If this value does not match the `actualStatusCode` value, the workflow step fails. This behavior is similar to what happens when a job template step returns a return code that is greater than the allowed maximum return code.	Required.	Integer	None
actualStatusCode	The actual HTTP status code that is received from the REST request. To obtain this value, map it to a workflow variable.	Optional	Integer	None
propertyMapping	The property from the REST response body that is mapped to a workflow variable. You can specify multiple `propertyMapping` elements in a REST step.	Optional	String	The following attributes are supported for the propertyMapping sub-element: **mapFrom** Specifies the supplied value to be mapped. This attribute is optional. If specified, the type must be nonNullString. **mapTo** Specifies the variable to which the supplied value is to be mapped. This attribute is optional. If specified, the type must be nonNullString.

Table 335. Step elements summary table: Additional sub-elements for steps that invoke another workflow (a called workflow)

Element name	Description	Required or optional	Type	Supported attributes
variableMapping	Used to transfer variable values between the called workflow and calling workflow.	Optional.	Any	No attributes are supported for this sub-element.
fromCallingToCalled	Used to transfer variable values from the calling workflow to the called workflow.	Optional.		No attributes are supported for this sub-element.
regExpression	Used for filtering on variable names. Specify a portion of the variable name with one or more wildcard characters.	Optional.	nonNullString	No attributes are supported for this sub-element.
variableName	Name of the variable.	Optional.	nonNullString	The following attribute is supported for the *variableName* sub-element: **mapTo** Specifies the variable to which the supplied value is to be mapped. This attribute is optional. If specified, the type must be nonNullString.
fromCalledToCalling	Used to transfer variable values from the called workflow to the calling workflow.	Optional		The following attribute is supported for the *fromCalledToCalling* sub-element: **override** Indicates whether the variable settings in this element are to take precedence over the variables in the calling workflow. The default is false. This attribute is optional. If specified, it must be a Boolean value.
regExpression	Used for filtering on variable names. Specify a portion of the variable name with one or more wildcard characters.	Optional.	nonNullString	No attributes are supported for this sub-element.
variableName	Name of the variable.	Optional.		No attributes are supported for this sub-element.
callingStepWeight	The relative difficulty of the step compared to other steps within this workflow. The *callingStepWeight* sub-element is mutually exclusive with the *step* sub-element.	Required.	Integer value 1 - 1000.	No attributes are supported for this sub-element.
callingStepSkills	The type of skills that are required to perform this step. The *callingStepSkills* sub-element is mutually exclusive with the *step* sub-element.	Optional.	nlsString	No attributes are supported for this sub-element.

Table 335. Step elements summary table: Additional sub-elements for steps that invoke another workflow (a called workflow) (continued)

Element name	Description	Required or optional	Type	Supported attributes
callingStepAutoEnable	Indicates whether the step is to be performed automatically when all prerequisite steps are completed, and no user inputs are required. If *callingStepAutoEnable* is not specified, the default is *false*.	Optional.	Boolean	No attributes are supported for this sub-element.
canCallingStepMarkAsFailed	Indicates whether the step can be marked as *Failed* manually by the step owner. If *canCallingStepMarkAsFailed* is not specified, the default is *false*. When set to *true*, the Review Instructions page in the Step Perform wizard includes the option to allow the step owner to mark a step as *Failed* manually. When *false*, this option is not displayed to the user.	Optional.	Boolean	No attributes are supported for this sub-element.
calledWorkflowDefinitionFile	Path name of an external file that contains the workflow definition for the called workflow. Either inlineTemplate or fileTemplate must be specified within the template sub-element.	Optional.	Path name to template file, the contents of which are treated as type velocityString.	No attributes are supported for this sub-element.
calledWorkflowDescription	A more detailed description of the step.	Required.	nlsRichString	No attributes are supported for this sub-element.
calledWorkflowID	The name of the workflow. The combination of calledWorkflowID and calledWorkflowVersion must be unique within the Workflows task. You can use the calledWorkflowID and calledWorkflowVersion to identify a called workflow.	Optional.	String consisting of letters (uppercase or lowercase), numeric digits, the hyphen, and the underscore character. This value must begin with a letter.	No attributes are supported for this sub-element.

Table 335. Step elements summary table: Additional sub-elements for steps that invoke another workflow (a called workflow) (continued)

Element name	Description	Required or optional	Type	Supported attributes
calledWorkflowVersion	The version of this workflow definition file. Update this value whenever you change any portion of the workflow definition file, including changes to the primary XML file or any sub-files or referenced files. The Workflows task caches only the latest version of an imported workflow definition file. Therefore, to ensure that the most current version is used, you must update the version value whenever you modify the workflow definition. For this reason, when you create a workflow definition file, you might want to complete the development phase on a workstation before you import the workflow definition into the Workflows task. The combination of calledWorkflowID and calledWorkflowVersion must be unique within the Workflows task. You can use the calledWorkflowID and calledWorkflowVersion to identify a called workflow.	Optional.	nonNullString	No attributes are supported for this sub-element.
calledWorkflowMD5	An MD5 encrypted value (a 128-bit hash value) that you can use to identify the called workflow.	Optional.	nonNullString	No attributes are supported for this sub-element.

Variable definition elements and types summary

Table 336. Variable definition elements summary

Element name	Description	Required or optional	Type	Supported Attributes
variable	Contains the definition of a variable. Up to 1500 variables can be defined in a workflow.	Optional.	If specified, the *variable* element can contain the following sub-elements: • label (required) • abstract (required) • description (required) • exposeToUser (optional) • category (required) • datastore (optional) And, the *variable* element must contain at least one of the type-specific sub-elements, which are listed in Table 337 on page 677.	The following attributes are supported for the *variable* element: **name** The name of the variable. The name is required, and must be a string consisting of letters (uppercase or lowercase), numeric digits, the hyphen, and the underscore character. This value must begin with a letter. The combination of name and scope must be unique within the workflow. **scope** The scope of the variable. The scope is required, and the value must be *instance* or *global*. The default is *instance*. The combination of name and scope must be unique within the workflow.
label	A short label for the UI widget.	Required if the variable element is specified.	nlsString	No attributes are supported for this sub-element.
abstract	A brief description of the variable for the UI widget.	Required if the variable element is specified.	nlsString	No attributes are supported for this sub-element.
description	A longer explanation of what the variable is used for, and perhaps what the syntactic requirements are.	Required if the variable element is specified.	nlsRichString	No attributes are supported for this sub-element.
exposeToUser	For a step that runs a JCL job. If included, the exposeToUser element indicates that the variable is to be included in the List variables for substitution window of the Workflows task. This element supports both global variables and instance variables in Workflows task. If this element is not specified, the variable is not shown in the Edit JOB statement page, and thus, cannot be selected by the user for the substitution of another value in the JOB statement.	Optional. Applicable only when the variable element is specified.	exposeToUserType	**usage** Specify the purpose of the variable. This text is displayed next to the variable in the Edit JOB Statement window of the Workflows task. The usage is optional. If specified, its data type must be an nlsString.
category	The name of the logical grouping to which this variable belongs. The default is *general*.	Required if the variable element is specified.	nlsString	No attributes are supported for this sub-element.

Table 336. Variable definition elements summary (continued)

Element name	Description	Required or optional	Type	Supported Attributes
datastore	Place where the variable value is stored over time. Because z/OSMF is the only supported location for variable values, it is not necessary to specify the datastore element.	Optional. Applicable only when the variable element is specified.	A single, empty, required *zOSMF* sub-element	No attributes are supported for this sub-element.

Table 337. Variable definition type-specific sub-elements summary

Element name	Description	Type	Supported attributes
Boolean	Boolean type	Empty	No attributes are supported for this sub-element.
default	Default value	Boolean. The default is *true*.	No attributes are supported for this sub-element.

Table 337. Variable definition type-specific sub-elements summary (continued)

Element name	Description	Type	Supported attributes
string	String type	If specified, this sub-element can contain one or more of the following sub-elements: **minLength** Minimum string length. This sub-element is optional. If specified, its value must be a non-negative integer. If the maxLength is specified, the maxLength must be greater than the minLength. The minLength and maxLength combination mutually excludes the validationType and regularExpression sub-elements. **maxLength** Maximum string length. This sub-element is optional. If specified, its value must be a non-negative integer. If the minLength is specified, the maxLength must be greater than the minLength. The minLength and maxLength combination mutually excludes the validationType and regularExpression sub-elements. **validationType** Validation types. This sub-element is optional. If specified, it must have one of the following values: ALPHA, ALPHAB, ALPHANUM, BIT, DSMEMBERNAME, DSNAME, DSQUAL, GROUP, HEX, IPADDR, IPADDR4, IPADDR6, TSOUSERID, UNIXID, USERID, VOLSER. The validationType sub-element mutually excludes the minLength and maxLength combination and the regularExpression sub-element. The UNIXID validation type verifies that a z/OS UNIX UID or GID is in the range 0 – 2147483647. Here, a UID or GID is treated as a string, not an integer. If you have code that treats a UID or GID as numeric, use an integer type to define the variable, instead of a string validation type. You can enforce the minimum and maximum values within the integer variable definition. **regularExpression** Standard regular expression that constrains the variable value. This sub-element is optional. If specified, it must have the bigNonNullString data type. The regularExpression sub-element mutually excludes the minLength and maxLength combination and the validationType sub-element. **errorMessage** Overrides the default error message for an incorrect value. This sub-element is optional. If specified, it must have an nlsString data type. **choice** Added to the choice list. This sub-element is required if *valueMustBeChoice=true*. Up to 1337 choices are allowed, and each choice must adhere to any minLength, maxLength, validationType, and regularExpression specified. **default** Widget that is primed with the value specified. This sub-element is optional. If specified, the default value must be an nlsUnboundedString that adheres to any minLength, maxLength, validationType, and regularExpression specified. If *valueMustBeChoice=true*, the default value must be one of the choice values specified.	The following attributes are supported for the *string* sub-element: **multiLine** Specifies a single-line text box widget or a multi-line text box. This attribute is optional, and its data type is Boolean. If unspecified, its value is *false* (default). **valueMustBeChoice** Indicates whether the value must come from the provided choices. This attribute is optional, and its data type is Boolean. If unspecified, its value is *false* (default). If set to true, at least one *choice* sub-element must be specified.

Table 337. Variable definition type-specific sub-elements summary (continued)

Element name	Description	Type	Supported attributes
integer	Integer type	If specified, this sub-element can contain one or more of the following sub-elements: **minValue** Minimum value. This sub-element is optional. If specified, its value must be an integer. If the maxValue is specified, the maxValue must be greater than or equal to the minValue. **maxValue** Maximum value. This sub-element is optional. If specified, its value must be an integer. If the minValue is specified, the maxValue must be greater than or equal to the minValue. **default** Widget that is primed with the value specified. This sub-element is optional. If specified, the default value must be an integer that adheres to any minValue and maxValue specified.	No attributes are supported for this sub-element.
decimal	Decimal type	If specified, this sub-element can contain one or more of the following sub-elements: **minValue** Minimum value. This sub-element is optional. If specified, its value must be a decimal. If the maxValue is specified, the maxValue must be greater than or equal to the minValue. **maxValue** Maximum value. This sub-element is optional. If specified, its value must be a decimal. If the minValue is specified, the maxValue must be greater than or equal to the minValue. **default** Widget that is primed with the value specified. This sub-element is optional. If specified, the default value must be a decimal that adheres to any minValue and maxValue specified. Decimal places are rounded by the Workflows task, based on the decimalPlaces attribute value.	The following attribute is supported for the *decimal* sub-element: **decimalPlaces** Maximum number of decimal places that can be specified. The value can be an integer in the range of 1 - 6. The default is 1.
time	Time type	If specified, this sub-element can contain one or more of the following sub-elements: **minValue** Minimum value. This sub-element is optional. If specified, its value must be the time in *hh:mm:ss* format. If the maxValue is specified, the maxValue must be greater than or equal to the minValue. **maxValue** Maximum value. This sub-element is optional. If specified, its value must be the time in *hh:mm:ss* format. If the minValue is specified, the maxValue must be greater than or equal to the minValue. **default** Widget that is primed with the value specified. This sub-element is optional. If specified, the default value must be the time in *hh:mm:ss* format and must adhere to any minValue and maxValue specified.	No attributes are supported for this sub-element.

Table 337. Variable definition type-specific sub-elements summary (continued)

Element name	Description	Type	Supported attributes
date	Date type	If specified, this sub-element can contain one or more of the following sub-elements: **minValue** Minimum value. This sub-element is optional. If specified, its value must be the date in *yyyy-mm-dd* format. If the maxValue is specified, the maxValue must be greater than or equal to the minValue. **maxValue** Maximum value. This sub-element is optional. If specified, its value must be the date in *yyyy-mm-dd* format. If the minValue is specified, the maxValue must be greater than or equal to the minValue. **default** Widget that is primed with the value specified. This sub-element is optional. If specified, the default value must be the date in *yyyy-mm-dd* format and must adhere to any minValue and maxValue specified.	No attributes are supported for this sub-element.
password	Password type.	If specified, this sub-element can contain one or more of the following sub-elements: **minLength** Minimum string length. This sub-element is optional. If specified, its value must be a non-negative integer. If the maxLength is specified, the maxLength must be greater than the minLength. The minLength and maxLength combination mutually excludes the regularExpression sub-element. **maxLength** Maximum string length. This sub-element is optional. If specified, its value must be a non-negative integer. If the minLength is specified, the maxLength must be greater than the minLength. The minLength and maxLength combination mutually excludes the regularExpression sub-element. **regularExpression** Standard regular expression that constrains the variable value. This sub-element is optional. If specified, it must have the bigNonNullString data type. The regularExpression sub-element mutually excludes the minLength and maxLength sub-elements. **errorMessage** Overrides the default error message for an incorrect value. This sub-element is optional. If specified, it must have an nlsString data type.	No attributes are supported for this sub-element.

atCreate element summary

Table 338. atCreate element summary

Element name	Description	Required or optional	Type	Supported Attributes
atCreate	For users of the Create Workflow REST service, the atCreate element provides additional options for working with variables. Up to 1500 atCreate elements can be defined in a workflow.	Optional.	atCreateType	The following attributes are supported for the *atCreate* element: **name** Specifies the variable for which the variable attributes are being set. This attribute is required. **scope** Specifies the scope of the variable. This value is set to *instance* (the only valid value) or is omitted; the default is *instance*. **required** For a workflow that is created through the Create Workflow REST service, this attribute indicates whether the variable must be set to a value at the time of workflow creation. This attribute is optional; the default is *false*. If a variable is marked as "required," but the variable is not given a value, an attempt to create the workflow through the Create Workflow REST service will fail with an error. For a workflow that is created through the Workflows task user interface, this option is ignored. **prompt** For users of the Create Workflow REST service, this attribute identifies a variable that *should* be prompted for by the program that issues the REST service. By itself, the prompt attribute does not enforce any behavior for the workflow creation. However, by setting prompt to *true*, you can indicate that prompting is recommended for the variable. The user of the Create Workflow REST service can query the value of the prompt attribute for any variables in the workflow to determine whether any variables should be prompted for. This attribute is optional; the default is *false*.

Chapter 3. Creating your own z/OSMF plug-ins

z/OSMF provides a modular framework that you can use to bring together all of your z/OS system management applications. z/OSMF supports different levels of integration ranging from adding a resource link or an application link to creating your own z/OSMF plug-ins.

To help you decide which integration method is best for you, consider the following recommendations:

- If your installation uses web-based applications that can launch or be launched by z/OSMF tasks or other web-based applications and display a specific context, it is recommended that you use the Application Linking Manager to create context-sensitive launch points between the applications.
- If your installation has commonly used web-based resources that do not have natural integration points with existing tasks or applications, it is recommended that you use the Links task to add the resource as a link in z/OSMF.
- If your installation requires function that is not provided in z/OSMF or in another web-based application, it is recommended that you create your own plug-in and use the Import Manager task to import the property file into z/OSMF.

Using these integration methods reduces context shifts between disparate applications, helps simply the management of your z/OS mainframe systems, and moves your installation one step closer to providing a central location for z/OS system management tasks.

The remainder of this section explains how to create your own z/OSMF plug-in. For more information about the Application Linking Manager task or the Links task, see the z/OSMF online help.

Process of creating a plug-in

In z/OSMF, a plug-in is a collection of one or more web-based applications (referred to as tasks) that add function to z/OSMF. z/OSMF ships with several plug-ins, which are described in the topic about z/OSMF system management tasks in *IBM z/OS Management Facility Configuration Guide*.

In addition to the shipped plug-ins, z/OSMF allows you to create your own plug-ins to add installation-specific function to z/OSMF. The process of adding your own plug-ins to z/OSMF includes the following activities:

1. Developing a web-based application and the supporting documentation for the functions you want to add to z/OSMF.
2. Storing the application and its documentation in the UNIX file system, and setting 644 permissions for files and 755 permissions for folders.
3. Creating a property file in the UNIX file system that defines the parameters required for z/OSMF to configure your plug-in.
4. Using the z/OSMF Import Manager task to import the property file.
5. Setting up security for your plug-in. After which, you must refresh the security management product on your system and restart the z/OSMF server to have your changes take affect.

For more details, see the sections that follow.

Developing web-based applications

The z/OSMF framework provides the infrastructure, security, and services you need to create the web-based applications to be included in your plug-ins.

Specifically, z/OSMF provides the following resources:

JavaScript APIs
z/OSMF core provides application programming interfaces (APIs) that allow you to create plug-ins that interact with z/OSMF core and share data with the tasks in your plug-ins. Exploiting these APIs provides users with a consistent experience across z/OSMF tasks. For more details, see "Using the z/OSMF core JavaScript APIs" on page 685 and "Using the Application Linking Manager JavaScript APIs" on page 700.

REST Services
z/OSMF provides several REST interfaces, which you can use to simplify the creation of your z/OSMF plug-ins. For more details, see Chapter 1, "Using the z/OSMF REST services," on page 1.

Client Side Logger
z/OSMF provides a client-side logging framework that you can use to route client messages to the z/OSMF log, which is the same log used by plug-ins that ship with z/OSMF. For more details, see "Logging client messages in the z/OSMF log" on page 712.

File Retrieval Service
z/OSMF provides a file retrieval service that you can use to specify the file for z/OSMF core to display when it launches your application and to specify any additional files or resources your application may need. For more details, see "Retrieving files and resources for your application" on page 719.

Secure Environment
z/OSMF uses the System Authorization Facility (SAF) interface on the z/OS host system to authenticate users and to grant users access to z/OS system management tasks. z/OSMF security also depends on plug-in developers identifying and remediating security vulnerabilities in their applications. For more details, see "Securing your applications" on page 741.

Requirements

Your application must:

- Be written in a markup or programming language that a Web browser can interpret, such as HTML, JavaScript, and CSS. z/OSMF does not support markup or programming languages that must be interpreted by the z/OSMF server.
- Use Dojo 1.7 or later to interface with z/OSMF core; however, you are not required to code your application using Dojo.
- Be stored in the UNIX file system with 644 permissions for files and 755 permissions for folders.
- Add new function to z/OSMF or update a plug-in that you created and previously imported. You cannot use plug-ins to modify the z/OSMF framework, the z/OSMF user interface, or other z/OSMF plug-ins.

Using the z/OSMF core JavaScript APIs

z/OSMF provides the zosmfExternalTools JavaScript API, which z/OSMF tasks can use to define actions to be performed before z/OSMF closes the task, to obtain a unique user session identifier, and to store and manage public objects in z/OSMF core.

The API is located at the following path: `/zosmf/js/zosmf/util`. To access the functions in the API and to dynamically add functions to the API, you must import and instantiate the zosmfExternalTools object in your task's HTML file. Sample code is provided in Figure 325.

```
require (["dojo/_base/window","zosmf/util/zosmfExternalTools"],
           function (win, zosmfExternalTools) {
          win.global.zosmfExternalTools = new zosmfExternalTools();
});
```

Figure 325. Sample JavaScript code for importing and instantiating the global zosmfExternalTools object

For information about the functions that are included in or can be added to the API, see the following sections:

- "Functions provided in the zosmfExternalTools API"
- "Functions you can add to the zosmfExternalTools API"

Functions provided in the zosmfExternalTools API

Table 339 lists the functions that are provided in the zosmfExternalTools API.

Table 339. Functions provided in the zosmfExternalTools API

Function	Usage	Where described
programmaticallyCloseTab	Call this function to request for z/OSMF core to close your task.	"programmaticallyCloseTab function" on page 691
cleanupBeforeDestroyComplete	Call this function to inform z/OSMF core that the actions performed by the cleanupBeforeDestroy function are complete. After which, z/OSMF closes your task.	"cleanupBeforeDestroyComplete function" on page 694
getUserSessionId	Call this function to retrieve the unique session identifier that z/OSMF core creates for each user and z/OSMF instance combination.	"getUserSessionId function" on page 695
definePublicObject	Call this function to define a public object in z/OSMF core.	"definePublicObject function" on page 696
retrievePublicObject	Call this function to retrieve a public object from z/OSMF core.	"retrievePublicObject function" on page 698
deletePublicObject	Call this function to delete a public object from z/OSMF core.	"deletePublicObject function" on page 699

Functions you can add to the zosmfExternalTools API

Table 340 on page 686 lists the functions you can define and dynamically add to the zosmfExternalTools API. z/OSMF core calls these functions when specific events are triggered.

Table 340. Functions you can add to the zosmfExternalTools API

Function	Usage	When called	Where described
isContentChanged	Define this function if you want your task to check for unsaved changes before z/OSMF closes the task.	z/OSMF core calls your isContentChanged function when: • A user clicks the X icon to close the z/OSMF tab that contains your task. • Your task calls the programmaticallyCloseTask function.	"isContentChanged function" on page 687
shouldClose	Define this function if you want to delay the close task request for an unspecified period of time so that your task can perform some additional actions.	z/OSMF core calls your shouldClose function when a user clicks the X icon to close the z/OSMF tab that contains your task.	"shouldClose function" on page 689
cleanupBeforeDestroy	Define this function if you want to delay the close task request for up to one second so that your task can perform some cleanup actions before z/OSMF closes the task.	z/OSMF core calls your cleanupBeforeDestroy function when a user: • Logs out of z/OSMF. • Clicks the X icon to close the z/OSMF tab that contains your task. • Closes the browser tab or window. • Changes the URL in the browser and redirects away from z/OSMF.	"cleanupBeforeDestroy function" on page 692

isContentChanged function

If your task allows users to save input values, modified content, or selections, consider defining an *isContentChanged* function that will check for unsaved changes before z/OSMF closes the task. Doing so prevents users from inadvertently discarding their work.

Invoking the function

The *isContentChanged* function must be defined off the zosmfExternalTools object. To define the *isContentChanged* function, use the syntax shown in Figure 326.

```
win.global.zosmfExternalTools.isContentChanged = function ( ) {
   //Define your function here.

   //Return either true or false.
   return true|false;
}
```

Figure 326. Syntax to use for the isContentChanged function

z/OSMF core calls the *isContentChanged* function provided for your task when a user clicks the X icon to close the tab that contains your task or when your task calls the programmaticallyCloseTask function.

Return values

The *isContentChanged* function returns a Boolean value, which indicates the following:

true Indicates that changes are pending. In this case, z/OSMF core displays a prompt that warns the user that unsaved changes will be discarded, and gives the user the option to proceed with closing the task or to cancel the close task request. Figure 327 depicts a sample prompt.

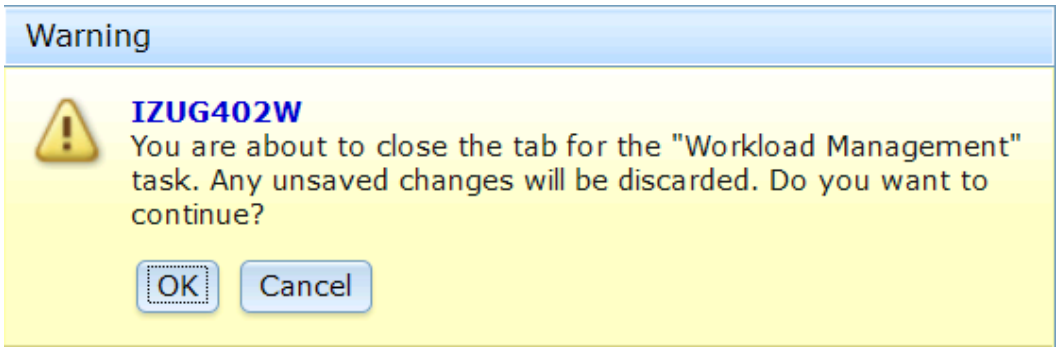

Figure 327. Sample confirmation window for a close task request

false Indicates that no changes are pending. In this case, z/OSMF core closes the task tab and does not display a prompt.

Note: If the zosmfExternalTools object or the *isContentChanged* function do not exist, z/OSMF core will display a prompt regardless of whether changes are pending.

Example

Suppose your plug-in contains a task that has multiple editable fields on a single page. When the user clicks the X icon to close your task tab, you want to check for changes for each field. Figure 328 on page 688 provides sample code you can use for this scenario.

```
//Define the default zosmfExternalTools.isContentChanged() function.
function init(){
   win.global.zosmfExternalTools = new zosmfExternalTools();
   win.global.zosmfExternalTools.isContentChanged = function (){
     return false;
   }
}

//When the page is created, call the internal _isContentChanged() function.
postCreate: function() {

   //Define the isContentChanged function.
   var that = this;
   win.global.zosmfExternalTools.isContentChanged = function (){
     return that._isContentChanged();
   }

},

//Determine if the content was changed.
_isContentChanged: function(){
   ...
},

//When the page is destroyed, restore the default isContentChanged function.
uninitialize: function() {
   win.global.zosmfExternalTools.isContentChanged = function (){
     return false;
   }

}
```

Figure 328. Sample code for the isContentChanged function

shouldClose function

When a close task request is submitted, if your task needs to perform cleanup actions or collect information from users before z/OSMF closes the task, consider defining a *shouldClose* function to override requests to close your task. Doing so gives your task an unlimited amount of time to perform the actions, and allows your task to inform z/OSMF when it is ready to be closed.

Invoking the function

z/OSMF core calls the *shouldClose* function provided for your task when a user clicks the X icon to close the tab that contains your task. The *shouldClose* function must be defined off the zosmfExternalTools object. To define the *shouldClose* function, use the syntax shown in Figure 329.

```
win.global.zosmfExternalTools.shouldClose = function ( ) {
   //Define your function here.

   //Return either true or false.
   return true|false;
}
```

Figure 329. Syntax to use for the shouldClose function

Return values

The *shouldClose* function returns a Boolean value, which indicates the following:

true Indicates that z/OSMF core should close the task.

false Indicates that z/OSMF core should not close the task at this time. The task will submit a close task request when it is ready to be closed. Return *false* if you need to perform additional steps or cleanup before closing the task. The task will remain open until your task calls the programmaticallyCloseTab function.

Tip: There is no time limit on how long the tab can remain open after a user requests for it to be closed. If the additional actions will take more than a few seconds to complete, consider providing an indicator so that users know the close request is being processed.

Note: If the zosmfExternalTools object or the *shouldClose* function do not exist, z/OSMF core will close the task.

Example

```
var functionToCall = function(){
  //Task tab is now open indefinitely.
  //Perform all cleanup work.
   ....
  //When its finished, call the programmaticallyCloseTab function.
  win.global.zosmfExternalTools.programmaticallyCloseTab(myPluginId,myTaskId);
}

win.global.zosmfExternalTools.shouldClose = function(){

  if(longCleanupNeeded){
  //If cleanup actions will take longer than one second,
  //call the shouldClose function, and sSet the cleanup
  //function to start asynchronously after the shouldClose
  //function returns false.
  setTimeout(functionToCall,5);
  return false;
  }else{
    //if we dont need more time and should close
    return true;
  }
}
```

programmaticallyCloseTab function

If your task overrode or intercepted a user's request to close your task, call the *programmaticallyCloseTab* function to request for z/OSMF core to close your task. Otherwise, your task will remain open.

Invoking the function

To call the *programmaticallyCloseTab* function, use the syntax shown in Figure 330.

```
win.global.zosmfExternalTools.programmaticallyCloseTab(pluginId, taskId);
```

Figure 330. Syntax to use to call the programmaticallyCloseTab function

where,

pluginId
: Unique identifier assigned to the plug-in that contains the task.

taskId Unique identifier assigned to the task that you want to close.

When your task calls this function, z/OSMF core looks up the *pluginId* and *taskId* and attempts to close the corresponding z/OSMF tab. This close task request has the same behavior as the close request that is submitted when a user clicks the X to close the task tab.

For this service to work, z/OSMF core must be the parent of the tab that contains your task. For example, z/OSMF can use the *programmaticallyCloseTab* function to close your task when your task is launched from the z/OSMF navigation tree or when the task is launched with context by the Application Linking Manager.

z/OSMF cannot use this service to close your task if the task is open in another browser tab or window or if a user made your task a link and launched the task directly.

Example

Suppose you have a plug-in with the ID *com.company.product*, and it contains a task with the ID *Product Name*. To close the task using the *programmaticallyCloseTab* function, you can use the sample code provided in Figure 331.

```
function closeMyself(){
   //Ensure that the Close Task confirmation window is not displayed.
   win.global.zosmfExternalTools.isContentChanged = function (){
      return false;
   }

   //Call the parent to refer to z/OSMF core. Then, call the function.
   parent.programmaticallyCloseTab("com.company.product","Product Name");
}
```

Figure 331. Sample code for the programmaticallyCloseTab function

cleanupBeforeDestroy function

When a close task request is submitted, if your task needs to perform cleanup actions before z/OSMF closes the task, such as ending a TSO/E address space, consider defining a *cleanupBeforeDestroy* function. Doing so delays a close task request for up to one second so that your task can perform the cleanup actions.

Invoking the function

The *cleanupBeforeDestroy* function must be defined off the zosmfExternalTools object. To define the *cleanupBeforeDestroy* function, use the syntax shown in Figure 332.

```
function init(){
   win.global.zosmfExternalTools = new zosmfExternalTools();

   //Define the cleanupBeforeDestroy function.
   win.global.zosmfExternalTools.cleanupBeforeDestroy = function (obj) {

      //Perform some cleanup
      ......


      //When cleanup is complete, call the cleanupBeforeDestroyComplete
      //function using the same parameter you used for the
      //cleanupBeforeDestroy function.
      win.global.zosmfExternalTools.cleanupBeforeDestroyComplete(obj);
   }

}
```

Figure 332. Syntax to use for the cleanupBeforeDestroy function

where,

obj Object that z/OSMF core creates to identify the task. You can specify any parameter name, but you must use the same name for the *cleanupBeforeDestroy* and *cleanupBeforeDestroyComplete* functions.

z/OSMF core calls the *cleanupBeforeDestroy* function provided for your task when a user:

- Logs out of z/OSMF.
- Clicks the X icon to close the z/OSMF tab that contains your task.
- Closes the browser tab or window.
- Changes the URL in the browser and redirects away from z/OSMF.

z/OSMF core calls the *cleanupBeforeDestroy* function every time one of the aforementioned events occurs. If a user has multiple instances of your task open or is working with multiple tasks in your plug-in, ensure that you do not release resources until the last task or the last instance of your task is closed.

After z/OSMF core calls your *cleanupBeforeDestroy* function, the task will remain open until you call the *cleanupBeforeDestroyComplete* function or until one second elapses, at which point, z/OSMF core will automatically close your task.

Tip: If you define a *cleanupBeforeDestroy* function, you do not need to define a *shouldClose* function because z/OSMF assumes *shouldClose* will return *false*.

Example

```
win.global.zosmfExternalTools.cleanupBeforeDestroy = function(obj) {

  //A synchronous cleanup method that cleans up objects on the client
  //to free up memory.
  myApi.cleanupMethod();

  //Inform z/OSMF core that the cleanup actions are complete
  //and that the task is ready to be closed.
  win.global.zosmfExternalTools.cleanupBeforeDestroyComplete(obj);
};
```

Figure 333. Sample code for the cleanupBeforeDestroy function

If myApi.cleanupMethod() is an asynchronous method, call the *cleanupBeforeDestroyComplete* function after the asynchronous action completes. Otherwise, the asynchronous method might not have time to complete.

cleanupBeforeDestroyComplete function

If your task defined a *cleanupBeforeDestroy* function, call the *cleanupBeforeDestroyComplete* function to inform z/OSMF core that your cleanup actions are complete and that your task is ready to be closed.

Invoking the function

To call the *cleanupBeforeDestroyComplete* function, use the syntax shown in Figure 334.

```
win.global.zosmfExternalTools.cleanupBeforeDestroyComplete(obj);
```

Figure 334. Syntax to use to call the cleanupBeforeDestroyComplete function

where,

obj Name of the parameter that was passed to the *cleanupBeforeDestroy* function.

The *cleanupBeforeDestroyComplete* function must be called after your *cleanupBeforeDestroy* function completes. Otherwise, z/OSMF might close the task before the cleanup actions are complete.

For example, if your *cleanupBeforeDestroy* function is performing an XMLHttpRequest (XHR) call, the *cleanupBeforeDestroyComplete* function must be called at the end of the XHR call and not immediately after the XHR call is issued.

Note: If your task does not call the *cleanupBeforeDestroyComplete* function within one second after z/OSMF invokes your *cleanupBeforeDestroy* function or if errors occur with the *cleanupBeforeDestroy* function, z/OSMF will close your task after one second elapses.

Example

```
win.global.zosmfExternalTools.cleanupBeforeDestroy = function(obj) {

  //A synchronous cleanup method that cleans up objects on the client
  //to free up memory.
  myApi.cleanupMethod();

  //Inform z/OSMF core that the cleanup actions are complete
  //and that the task is ready to be closed.
  win.global.zosmfExternalTools.cleanupBeforeDestroyComplete(obj);
};
```

Figure 335. Sample code for the cleanupBeforeDestroyComplete function

If myApi.cleanupMethod() is an asynchronous method, call the *cleanupBeforeDestroyComplete* function after the asynchronous action completes. Otherwise, the asynchronous method might not have time to complete.

getUserSessionId function

z/OSMF core creates a unique ID for the authenticated user for every browser tab or window in which the user has z/OSMF or a z/OSMF task opened. Your task can use the ID for any purpose. To retrieve the ID, call the *getUserSessionId* function.

Invoking the function

To call the *getUserSessionId* function, use the syntax shown in Figure 336.

```
win.global.zosmfExternalTools.getUserSessionId();
```

Figure 336. Syntax to use to call the getUserSessionId function

The session ID has the form *userName_timestamp*, where *userName* is the ID the user used to log into z/OSMF, and the *timestamp* is the date and time the user logged into z/OSMF or opened z/OSMF or a z/OSMF task in a new browser tab or window. The timestamp is based on the locale and time zone setting for the user's browser.

Example

```
var userSessionID;
function init(){
  //Create a unique session ID for the session to track if multiple
  //instances of the application are open.
  userSessionID=win.global.zosmfExternalTools.getUserSessionId();

  //This session ID can be used in any client side logging.
}
```

Figure 337. Sample code for the getUserSessionId function

definePublicObject function

In z/OSMF, a public object is a Dojo AMD class that IBM, you, or a vendor defined that has been instantiated and stored in z/OSMF core, making the class accessible to any vendor and any z/OSMF task. If your plug-in contains multiple tasks and you want to pass data between those tasks, consider creating a public object for the tasks to share. To do so, call the *definePublicObject* function.

For information about creating Dojo classes, see the following web pages:

- http://dojotoolkit.org/documentation/tutorials/1.7/declare/
- http://dojotoolkit.org/reference-guide/1.7/dojo/declare.html

Invoking the function

To call the *definePublicObject* function, use the syntax shown in Figure 338.

```
win.global.zosmfExternalTools.definePublicObject(class,handle,callback[,
    dojoPackageAlias,dojoPackageLocation]);
```

Figure 338. Syntax to use to call the definePublicObject function

where,

class Specify either the path to the Dojo class or the object of the Dojo class that you want to publicly store in z/OSMF core. When specifying the path, you can specify a path that is relative to the Dojo package location, or you can specify the absolute path. The class is required.

handle
Specify the name you want z/OSMF core to assign to the public object. The handle is required.

callback
Specify the function you want z/OSMF core to call when the object is available for your plug-in to use. The callback function is required.

dojoPackageAlias
Specify the shortcut or alias to use when referring to the location of the package that contains the Dojo classes. z/OSMF core will map the alias to the actual location of the package. The Dojo package alias is required only if you specify a path for the **class** parameter.

dojoPackageLocation
Specify the absolute path to the Dojo package. z/OSMF core will map this path to the Dojo package alias you specified. The Dojo package location is required only if you specify a path for the **class** parameter.

When your plug-in calls the *definePublicObject* function, if a path is specified for the **class** parameter, z/OSMF core:

1. Resolves the alias, and locates the Dojo package.
2. Instantiates the Dojo class, turning it into an object.
3. Stores the object in z/OSMF core under the specified handle.
4. Calls the callback function, which you provided, and passes the object to the function for your plug-in to use.

When your plug-in calls the *definePublicObject* function, if an object of the Dojo class is specified for the **class** parameter, z/OSMF core:

1. Stores the object in z/OSMF core under the specified handle.
2. Calls the callback function, which you provided, and passes the object to the function for your plug-in to use.

Example

In the following example, the path to the Dojo package is `/zosmf/IzuUICommon/externalfiles/util/js`. The alias for this path is `zosmfUtil`.

The path to the class that will be defined as a public object is `/zosmf/IzuUICommon/externalfiles/util/js/UtilConstants`. With the alias, the path to the class becomes `zosmfUtil/UtilConstants`. The public object will be called `utilGlobal`.

After z/OSMF core creates a new instance of the public object, z/OSMF core calls the testCallback function and supplies the handle for the object.

```
var testCallback = function(obj){
   console.log("callback called!");
   console.log("with object "+obj);
}

function init(){
win.global.zosmfExternalTools.definePublicObject(
   "zosmfUtil/UtilConstants","utilGlobal",testCallback,
   "zosmfUtil","/zosmf/IzuUICommon/externalfiles/util/js");
}
```

Figure 339. Sample code for the definePublicObject function

retrievePublicObject function

To retrieve a public object that is stored in z/OSMF core, call the *retrievePublicObject* function.

If a plug-in contains multiple tasks, for the first task opened during a z/OSMF session, call the *definePublicObject* function to create the object. When subsequent tasks in the plug-in are opened, call the *retrievePublicObject* function to obtain an instance of the object that has already been created.

If the *retrievePublicObject* function returns an empty string, the object has not been created in z/OSMF core. This may be an indicator that the calling task needs to define the public object.

Invoking the function

To call the *retrievePublicObject* function, use the syntax shown in Figure 340.

```
var publicObject=win.global.zosmfExternalTools.retrievePublicObject(handle);
```

Figure 340. Syntax to use to call the retrievePublicObject function

where *handle* is the handle used when the object was created.

Example

```
var util=win.global.zosmfExternalTools.retrievePublicObject("utilGlobal");
console.log("utilGlobal "+utilGlobal);
```

Figure 341. Sample code for the retrievePublicObject function

deletePublicObject function

To delete a public object that is stored in z/OSMF core, call the *deletePublicObject* function. After you delete a public object, it is not retrievable through the *retrievePublicObject* function because the object is no longer stored in z/OSMF core.

Invoking the function

To call the *deletePublicObject* function, use the syntax shown in Figure 342.

```
win.global.zosmfExternalTools.deletePublicObject(handle,param);
```

Figure 342. Syntax to use to call the deletePublicObject function

where,

handle
Handle used when the object was created.

param JSON object array that contains the name and value for each parameter that z/OSMF core will pass to the object's destroy() method. Specifying parameters is optional. The syntax to use follows:

```
{parm1: value1, parm2: value2, parm3: value3}
```

When your plug-in calls the *deletePublicObject* function, z/OSMF core:

1. Uses the handle to retrieve the public object.
2. Cleans up the public object.
3. Calls the destroy() method if the method is defined in the object. The destroy() method is used to further clean up the object.
4. Passes any parameters to the destroy() method.

The public object is also deleted when a user:

- Logs out of z/OSMF.
- Clicks the X icon to close the z/OSMF tab that contains your task.
- Closes the browser tab or window.
- Changes the URL in the browser and redirects away from z/OSMF.

Example

```
win.global.zosmfExternalTools.deletePublicObject("utilGlobal");
```

Figure 343. Sample code for the deletePublicObject function

Using the Application Linking Manager JavaScript APIs

If your installation uses multiple, disparate Web interfaces to manage your z/OS systems, use the z/OSMF Application Linking Manager to connect the applications. Doing so allows one task or application -- an event requestor -- to request that specific function or context be launched in another task or application -- the event handler, providing a smoother transition between applications.

z/OSMF provides the following resources for working with the Application Linking Manager:

- **Application Linking Manager task**, which provides a graphical user interface that you can use to add, query, or remove event type and event handler definitions.
- **Application Linking Manager REST APIs**, which are a set of REST services that allows a client application to add, query, or remove event type and event handler definitions.
- **AppLinker JavaScript API**, which is a set of JavaScript services that allows a client application to send events to the Application Linking Manager or to define the context to be displayed. The JavaScript services are applicable only if you are creating your own z/OSMF plug-in.

The remainder of this section describes the AppLinker JavaScript API. For information about the Application Linking Manager task, see the z/OSMF online help. For details about the REST APIs, see "Application Linking Manager interface services" on page 5.

z/OSMF predefines several event types, requestors, and handlers. For a list, see "Event types, requestors, and handlers shipped with z/OSMF" on page 7.

Importing and instantiating the AppLinker API

To participate in the application linking process as an event requestor or an event handler, your task needs access to the functions provided in the AppLinker API, which is located at the following path: `/zosmf/js/zosmf/izual`.

To access the functions in the API, you must import and instantiate the AppLinker API in your task's HTML file. Sample code is provided in Figure 344.

```
//Add a package for izual
packages: [{
   name: "izual",
   location: "/zosmf/js/zosmf/izual"
}]

//Import the AppLinker API.
require(["izual/api/PluginAppLinker17"],
         function(PluginAppLinker){
}
//Instantiate the global AppLinker variable.
win.global.applinker = new PluginAppLlinker();

//After instantiation, call the zOSMFTools getAppLinker function
win.global.zosmfTools=new Tools();
var localAppLinkerVariable = win.global.zosmfTools.getAppLinker();
```

Figure 344. Sample code for importing and instantiating the AppLinker API

Functions provided in the AppLinker API

Table 341 lists the functions that are provided in the AppLinker API.

Table 341. Functions provided in the AppLinker API

Function	Usage	Where described
sendEvent	If your task is an event requestor, call this function to send an event to the Application Linking Manager.	"sendEvent function" on page 703
getHandlers	If your task is an event requestor, call this function to determine if handlers are available to process an event.	"getHandlers function" on page 704
hasLaunchContext	If your task is an event handler and supports the *launch with context* or *launch with context and reload* launching option, call this function to determine if your task is being loaded as a result of an application linking event.	"hasLaunchContext function" on page 706
getEventFromUrl	If your task is an event handler and the hasLaunchContext function returns *true*, call the getEventFromUrl function to retrieve the event information that was supplied with the event.	"getEventFromUrl function" on page 707
subscribe	If your task is an event handler and supports the *launch with context and switch* launching option, call this function to define a JavaScript function that z/OSMF core will call when an event of the specified type is delivered to your task.	"subscribe function" on page 709
onLoadingComplete	If your task is an event handler and supports the *launch with context and switch* launching option, after your task subscribes to the event types it can handle, call this function to inform the Application Linking Manager that your task is ready to handle events.	"onLoadingComplete function" on page 711

Using the AppLinker functions in the application linking process

The application linking process consists of the following steps:

1. An event requestor defines a user interface control that invokes the *sendEvent* function when a user performs an action.
2. An event requestor calls the *getHandlers* function to determine if handlers are available to process the request. If handlers are not available, the event requestor might perform an action such as disabling or hiding the user interface control.
3. A user performs an action on the user interface control that triggers the call of the *sendEvent* function.
4. The Application Linking Manager searches the set of known event types for the type identified by the event.
5. If a match is found, the Application Linking Manager searches for event handlers that are registered for this event type. If only one handler is found, it is launched. Otherwise, the user is prompted to select the handler to launch.
6. After the handler is identified, z/OSMF core identifies the launch context the handler supports.

7. If the launch context is *launch without context*, z/OSMF core uses the URL provided in the handler definition to launch the handler. If the handler is already open, it receives focus.
8. If the launch context is *launch with context*, *launch with context and reload*, or *launch with context and switch*, z/OSMF core does the following:
   - Appends the event type and parameters to the URL provided in the handler definition.
   - Completes one of the following actions:
     - If the launch context is *launch with context*, z/OSMF core uses the modified URL to launch the handler. If the handler is already open, it receives focus.
     - If the launch context is *launch with context and reload*, z/OSMF core uses the modified URL to launch the handler. If the handler is already open, a message is displayed warning the user that the current context will be overwritten.
     - If the launch context is *launch with context and switch*, the following steps are completed:
       a. z/OSMF core uses the modified URL to launch the handler.
       b. If the handler is loading for the first time, the handler calls the *subscribe* function for each event type to which it wants to subscribe and provides the function that z/OSMF core will call to determine the context to display.
       c. After the handler subscribes to all the event types it can process, the handler calls the *onLoadingComplete* function to inform z/OSMF core that it is ready to accept events.
       d. When an event occurs, z/OSMF core verifies that the handler has called the *onLoadingComplete* function. If the function has been called, z/OSMF core searches the list of handlers that have subscribed for this event type, and identifies the callback function for the selected handler.

          Important: Register your task for each event type to which your task subscribes. Otherwise, z/OSMF core will not use the subscription because users will not have the option of selecting your task as the handler for the event.
       e. z/OSMF core calls the *subscribe* callback function.
       f. The function returns the context for z/OSMF core to display.
       g. z/OSMF core displays the context and finishes loading the handler.
9. To display the correct context for the *launch with context* and *launch with context and reload* launching options, the handler must do the following while it is being loaded:
   a. Call the *hasLaunchContext* function to determine if an event has occurred.
   b. Call the *getEventFromUrl* function to extract the event information from the URL that z/OSMF used to launch the handler.
   c. Use the event information to display the correct context. For example, the handler might call another JavaScript function, which you provide, that can process the event information and return the correct context.

sendEvent function

If your task is an event requestor, call the *sendEvent* function to send an event to the Application Linking Manager. The *sendEvent* function initiates the application linking process.

Overview

The *sendEvent* function is typically attached to a user interface control, such as a link, a button, or an action. When the user interacts with this control, the task calls the *sendEvent* function, which supplies the event type ID and parameters to the Application Linking Manager. The Application Linking Manager performs several actions to identify the handler to be launched, and then it launches the handler.

To complete the application linking process, the event type specified for the *sendEvent* must be registered with the Application Linking Manager, and at least one handler must be available to process the request.

To minimize errors during the application linking process, call the *getHandlers* function and verify that at least one handler is available to handle the event. If no handlers are available, consider hiding or disabling the user interface control that calls the *sendEvent* function.

Invoking the function

To call the *sendEvent* function, use the syntax shown in Figure 345.

```
win.global.applinker.sendEvent(eventTypeId, params);
```

Figure 345. Syntax to use to call the sendEvent function

where,

eventTypeId
ID that identifies the type of event.

params
JSON object array that contains the name and value for each parameter that your task will provide with the event. Specifying parameters is optional. The syntax to use follows:

```
{parm1: value1, parm2: value2, parm3: value3}
```

Example

```
win.global.applinker.sendEvent("IBM.ZOSMF.VIEW_DATASET",{"dataSetName":"myDataSet"});
```

Figure 346. Sample code for the sendEvent function

getHandlers function

If your task is an event requestor, consider calling the *getHandlers* function to determine if handlers are available to process your request. The *getHandlers* function does not initiate application linking. It helps your task determine if application linking is possible.

Overview

The *getHandlers* function identifies handlers that satisfy the following criteria:

- The handler is registered as a handler for the event type.
- The handler is enabled for the event type.
- The user is authorized to access the handler.

If one or more handlers satisfy this criteria, application linking is possible. Otherwise, application linking is not possible. In the latter case, consider hiding or disabling the user interface control that will initiate the application linking process. Doing so increases the usability of your task because the control is enabled or displayed only when the Application Linking Manager can successfully process the user's request.

Tip: If the *getHandlers* function does not find handlers that satisfy the aforementioned criteria, ensure that the event type is registered with the Application Linking Manager. If the event type is registered, verify that the event type ID is spelled correctly in your task.

Invoking the function

To call the *getHandlers* function, use the syntax shown in Figure 347.

```
//Define a function for the getHandlers function to call if the request
//completes without errors.
var callback = function(response){
      //Specify what to do if there are handlers.
   }
   else {
      //Specify what to do if there are no handlers.
   }
}

//Define a function for the getHandlers function to call if errors
//occur with the request.
var errback = function(error){
   //Specify how to proceed.
}

//Call the getHandlers function.
win.global.applinker.getHandlers(eventTypeId,callback,errback);
```

Figure 347. Syntax to use to call the getHandlers function

where,

response
JSON object array, which is provided by the *getHandlers* function, that contains the name of each handler that is available to process the event. To access the handlers in the array, use `response.results`.

errors JSON object array, which is provided by the *getHandlers* function, that contains the error messages the function received. To access the messages in the array, use `error.messages`.

eventTypeId
ID that identifies the type of event.

callback

Function, which you provide, that the *getHandlers* function will call if it completes without errors.

errback

Function, which you provide, that the *getHandlers* function will call if errors occur when it is processing the request.

Example

```
var callback = function(response){
   if(response && response.results && response.results.length>0){
       //Handlers exist; enable the user controls.
   }else{
      //No handlers exist; disable the user controls.
   }
}
var errback = function(error){
   //Error occured when retrieving handlers; retrieve the messages.
   var messages = error.message;
}
win.global.applinker.getHandlers("IBM.ZOSMF.VIEW_DATASET",callback,errback);
```

hasLaunchContext function

If your task is an event handler, when your task is loading, call the *hasLaunchContext* function to determine if your task is being launched as a result of an application linking event. Doing so allows your task to determine if it needs to collect event information and display a specific context, or if it can display the main page.

Invoking the function

Call the *hasLaunchContext* function only if the launching option specified for your task in the handler definition is *launch with context* or *launch with context and reload*. For the *launch without context* launching option, z/OSMF core does not collect event information because there is no context to display; therefore, it is not necessary for your task to call the *hasLaunchContext* function.

For the *launch with context and switch* launching option, z/OSMF core alerts your task that an event has occurred when it calls the callback function you specified for the subscribe function; therefore, it is not necessary for your task to call the *hasLaunchContext* function to determine if an event has occurred.

To call the *hasLaunchContext* function, use the syntax shown in Figure 348.

```
win.global.applinker.hasLaunchContext();
```

Figure 348. Syntax to use to call the hasLaunchContext function

Return values

The *hasLaunchContext* function returns a Boolean value, which indicates the following:

true Indicates that the Application Linking Manager delivered an event to the task. Call the *getEventFromUrl* function to parse the URL that z/OSMF core used to launch the task and retrieve the event information.

false Indicates that the Application Linking Manager has not delivered an event to the task. In this case, z/OSMF core will launch the task without context.

Example

```
function init(){
{
  if(win.global.applinker.hasLaunchContext()){
  //The task was opened with application linking; therefore, update context.

  //Use the getEventFromURL function to obtain the context.
  var result = win.global.applinker.getEventFromURL();
  var eventType = result.type;
  var params = result.params;

  //Use the eventType and parameters to switch the context.

  }
  else{
  //The task was not opened with application linking; therefore, display
  //the standard starting page.

  }
}
```

Figure 349. Sample code for the hasLaunchContext function

getEventFromUrl function

If your task is an event handler and the *hasLaunchContext* function returns *true*, call the *getEventFromUrl* function to parse the URL that z/OSMF core used to launch your task. Doing so will provide your task with the event type ID and parameters it needs to display the correct context. Otherwise, your task will launch without context.

Invoking the function

To call the *getEventFromUrl* function, use the syntax shown in Figure 350.

```
win.global.applinker.getEventFromUrl();
```

Figure 350. Syntax to use to call the getEventFromUrl function

Return values

The *getEventFromUrl* function returns an event object that includes the following information:

type The event type ID supplied with the event.

params
Array that contains the name and value of each parameter supplied with the event.

Tip: While your task is loading, it must use the event information to display the correct context. For example, your task might call another JavaScript function, which you provide, that can process the event information and return the correct context.

Expected results

The event information returned by the *getEventFromUrl* function depends on the launching option that is specified in the event handler definition for your task and whether your task is already open when an event is received.

Table 342 describes the expected result for each launching option and task state (open or closed) combination.

Table 342. Expected results by launching option and task state

Launching option and task state	Expected result
The launching option is *launch without context* and the task is closed.	The *getEventFromUrl* function will return *null*, and z/OSMF core will launch the task using the URL supplied in the handler definition. **Tip:** z/OSMF core does not collect event information for this launching option because there is no context to display; therefore, it is not necessary for your task to call the *getEventFromUrl* function.
The launching option is *launch without context* and the task is already open.	The *getEventFromUrl* function will return *null*, and z/OSMF core will bring the existing task tab into focus. **Tip:** z/OSMF core does not collect event information for this launching option because there is no context to display; therefore, it is not necessary for your task to call the *getEventFromUrl* function.
The launching option is *launch with context* and the task is closed.	The *getEventFromUrl* function will return the event type ID and parameters for the current event, and the task will display the current context.

Table 342. Expected results by launching option and task state (continued)

Launching option and task state	Expected result
The launching option is *launch with context* and the task is already open.	The *getEventFromUrl* function will return one of the following values: • The event type ID and parameters for the event that z/OSMF core used to initially open the task. • *Null* if the task was not initially opened as a result of an application linking event. z/OSMF core will bring the existing task tab into focus, and the task will display its last context.
The launching option is *launch with context and reload* and the task is closed.	The *getEventFromUrl* function will return the event type ID and parameters for the current event, and the task will display the current context.
The launching option is *launch with context and reload* and the task is already open.	z/OSMF core brings the existing tab into focus and displays a message, which warns the user that the current context will be overwritten and gives the user the option to proceed with displaying the new context or keeping the previous context. If the user clicks **OK**, the *getEventFromUrl* function will return the event type ID and parameters for the new event, and the task will display the new context. If the user clicks **Cancel**, the *getEventFromUrl* function will return one of the following values, and the task will display its last context: • The event type ID and parameters for the previous event. • *Null* if no event has occurred.
The launching option is *launch with context and switch* and the task either is closed or is already open.	The *getEventFromUrl* function will return the event information for the current event, and your task will determine the context to display based on the callback function you provided as a parameter for the *subscribe* function. **Tip:** z/OSMF core supplies your task with the event type ID and event parameters when it calls the *subscribe* callback function; therefore, it is not necessary for your task to call the *getEventFromUrl* function to retrieve the event information. For information about the *subscribe* function, see "subscribe function" on page 709.

Example

```
function init(){
{
  if(win.global.applinker.hasLaunchContext()){
  //The task was opened with application linking; therefore, update context.

  //Use the getEventFromURL function to obtain the context.
  var result = win.global.applinker.getEventFromURL();
  var eventType = result.type;
  var params = result.params;

  //Use the eventType and parameters to switch the context.

  }
  else{
  //The task was not opened with application linking; therefore, display
  //the standard starting page.

  }
}
```

Figure 351. Sample code for the getEventFromUrl function

subscribe function

If your task is an event handler and supports the *launch with context and switch* launching option, you must define the context for your task to display. To do so, your task must call the *subscribe* function for each event type it can handle and provide a JavaScript function for z/OSMF core to call when the Application Linking Manager delivers an event of the specified type.

Difference between subscribing and registering for an event type

Subscribing your task to an event type is different than registering your task as a handler for an event type. Registration is done regardless of launch context and it informs z/OSMF core that your task can handle events of the specified type. As such, z/OSMF core will present your task to users as a possible handler for events of that type.

Subscriptions are supported for the *launch with context and switch* launching option only, and it tells z/OSMF core how to process events of the specified type.

For the *launch with context and switch* launching option, your task must register for and subscribe to an event type. If your task subscribes but does not register as a handler for an event type, z/OSMF core will not use your task's subscription because it is impossible for users to select your task as the handler for the event type. If your task registers but does not subscribe to an event type, z/OSMF core will launch your task without context.

Invoking the function

Your task must call the *subscribe* function while it is still loading so that when it is loaded the correct context is shown. To call the *subscribe* function, use the syntax shown in Figure 352.

```
//Define a function that specifies how to handle the event.
var subscribeCallback = function(params, eventData) {
  //Define your callback function here.
}

//Call the subscribe function.
win.global.applinker.subscribe(eventTypeId, subscribeCallback);
```

Figure 352. Syntax to use to call the subscribe function

where,

params
: Object that contains an array of the parameters supplied with the event. z/OSMF core will provide this object when it calls your callback function.

eventData
: Object that contains the event type ID. z/OSMF core will provide this object when it calls your callback function.

eventTypeId
: ID of the event type to which your task is subscribing.

subscribeCallback
: Function, which you provide, that specifies the context for your task to display when z/OSMF core delivers an event with the specified ID to your task.

  Tip: If your subscribeCallback function is a function in your current object, you might be required to use dojo.hitch(this,"subscribeCallback") to specify it in the *subscribe* function call.

To subscribe to multiple event types, your task must call the *subscribe* function for each event type and, more than likely, provide a unique subscribe callback function for each call. For example, if your task is subscribing to three event types, it will call the subscribe function three times and, possibly, provide three unique callback functions.

After your task subscribes to all the event types it can handle, your task must call the *onLoadingComplete* function to inform z/OSMF core that it is ready to accept events. If your task does not call the *onLoadingComplete* function, z/OSMF core will not deliver the event that initially launched your task. z/OSMF core will, however, deliver subsequent events if your task has subscribed to the corresponding event type.

When z/OSMF core delivers an event to your task, z/OSMF core searches the list of tasks that have subscribed for the corresponding event type and identifies the callback function for your task. Then, z/OSMF core calls the callback function, and displays the context the function returns.

Tip: Your task does not have to unsubscribe the callback functions because z/OSMF core automatically unsubscribes the callback functions when your task uninitializes the AppLinker API.

Example

```
var viewDataset = function(response){
  //View the parameters in the response, and perform the context switching.
}

function init(){
    win.global.applinker.subscribe("IZU.ZOSMF.VIEW_DATASET",viewDataset);
}
```

Figure 353. Sample code for the subscribe function

onLoadingComplete function

If your task is an event handler and supports the *launch with context and switch* launching option, after your task subscribes to the event types it can handle, call the *onLoadingComplete* function to inform z/OSMF core that your task is ready to receive events.

Invoking the function

To call the *onLoadingComplete* function, use the syntax shown in Figure 354.

```
win.global.applinker.onLoadingComplete(suppressInitialEvents);
```

Figure 354. Syntax to use to call the onLoadingComplete function

where, suppressInitialEvents is a Boolean variable that indicates the following:

true Indicates that z/OSMF core will suppress the event that was used to initially launch your task. That is, z/OSMF core will ignore the first event, but will deliver subsequent events to the task.

Tip: Consider setting the suppressInitialEvents variable to *true* if you always want your task to load a certain way. For example, you might opt to set this variable to *true*, if you want your task to display a warning message or other important information that users must review before proceeding with your task.

If you want your task to process the initial event at a later time, assuming that no subsequent events have been received, you can retrieve the event information by calling the *getEventFromUrl* function and pass that information to the callback function that z/OSMF core would have called.

false Indicates that z/OSMF core will deliver the event that was used to initially launch your task so that your task can display the correct context.

Tip: If you want to set the suppressInitialEvents variable to *false*, you can use the syntax that is provided in either of the following options:

```
//Option 1:
call win.global.applinker.onLoadingComplete(false);

//Option 2:
call win.global.applinker.onLoadingComplete();
```

Example

```
function init(){
   setupMyPlugin();
}

function setupMyPlugin(){
    //Perform setup actions.
    ...
    //Call the onLoadingComplete function to indicate that loading is
    //complete so the task can switch context.
    win.global.applinker.onLoadingComplete(false);
}
```

Logging client messages in the z/OSMF log

z/OSMF provides a client side logging framework, which you can use to log your plug-in's client messages in the z/OSMF log. Using the client side logger helps simplify debugging activities because messages for all z/OSMF plug-ins are included in the same log.

For more information about the z/OSMF log, including how to access and view the log, see the topic about working with z/OSMF runtime log files in *IBM z/OS Management Facility Configuration Guide*.

Setting up client side logging

To implement client side logging, add code to your plug-in that:

- Specifies the name and location of the Dojo package that contains the client side logger.
- Imports the client side logger package.
- Creates an instance of the client side logger, and assigns a unique identifier to the logger. To uniquely identify your plug-in's client side logger, include the name of your company, product, and module in the identifier, and specify "ui" as the lower level qualifier. For example, `com.ibm.zosmf.NavigationTree.ui.`
- Defines the methods for the logging levels to be included in the z/OSMF log. The logging levels include: FINE, FINER (trace), FINEST (debug), INFO, WARNING, and SEVERE (fatal).

  To help you identify your plug-in's messages in the log, append a prefix to each message that contains the module name, package name, and method name.

```
var LOGPREFIX=MODULE+" "+PACKAGE_NAME+" "+methodName+": ";
LOGGER.entering(LOGPREFIX+"entry stuff");
```

Figure 355. Sample code for creating a log prefix

  For sample log data, see “Sample z/OSMF client side log data” on page 716.

- Issues JavaScript log statements that match the logging levels you specified. When issued, the client side logger stores the statements in a message queue until the interval, threshold, or trigger is activated, as follows:

  interval
  : By default, the client side logger sends the queued messages to the z/OSMF log every 60 seconds.

  threshold
  : By default, the message queue can contain a maximum of 50 JavaScript log statements. When this threshold is reached, the client side logger sends the queued messages to the z/OSMF log.

  trigger
  : By default, the push level for the client side logger is INFO. When the severity of a JavaScript log statement equals or exceeds the push level, the client side logger sends the queued messages to the z/OSMF log.

  Note: To prevent denial of service attacks or filling the log with useless data, the client side logger logs only messages that your plug-in issues on behalf of z/OSMF authenticated users.

Sample code for setting up client side logging

Figure 356 on page 713 provides sample code that you can use as a reference when setting up the client side logger.

```
var dojoConfig = {
isDebug:      false,
parseOnLoad:  false,
usePlainJson: true,
async: true,

//Specify the name and location of the client side logger package.
packages: [{name: "izuLogger", location: "/zosmf/IzuUICommon/js"}]
};

//Import the client side logger package.
require(["izuLogger/izuUILogger/log4js17"],

function(log4js){
    //Create an instance of the client side logger.
    var LOGGER = log4js.getLogger("com.mycompany.myproduct.mymodule.ui");
    var MODULE = "MYMODULE";
    var PACKAGE_NAME = "MYPACKAGE.jsp";

    //Define methods for logging messages.
    function init(){
        var methodName="init";
        var LOGPREFIX=MODULE+" "+PACKAGE_NAME+" "+methodName+": ";
        LOGGER.entering(LOGPREFIX+"entry stuff");
        LOGGER.finest(LOGPREFIX+"finest stuff");
        LOGGER.finer(LOGPREFIX+"finer stuff");
        LOGGER.info(LOGPREFIX+"info stuff");
        LOGGER.warning(LOGPREFIX+"warning stuff");
        LOGGER.severe(LOGPREFIX+"severe stuff");
        LOGGER.exiting(LOGPREFIX+"exiting stuff");
    }
});
```

Figure 356. Sample code for setting up client side logging

Methods provided for the client side logger

Table 343 on page 714 lists the methods you can use when setting up the client side logger, and provides summary descriptions and sample JavaScript code for each method.

Table 343. Methods provided for the client side logger

Method	Description	Sample Code
log4js.getLogger(*logger-ID*);	Creates an instance of the client side logger for your plug-in, and assigns a unique identifier (*logger-ID*) to the logger.	`var LOGGER = log4js.getLogger(` `"com.ibm.zosmf.NavigationTree.ui");`
loggerName.*log-level*(*param*);	Defines the log levels (*log-level*) you want the specified logger (*loggerName*) to include in the z/OSMF log, and takes a single string parameter (*param*) that specifies the information to log for each message. For the *loggerName*.severe(*param*) method, the client side logger automatically flushes messages to the z/OSMF log.	`LOGGER.warning(LOGPREFIX+"Test warning message");` `LOGGER.severe(LOGPREFIX+" Exception: " + text);`
loggerName.entering(*params*);	Logs the entry point of each method, and takes zero or more comma-separated parameters (*params*).	`LOGGER.entering(LOGPREFIX+"entry stuff");`
loggerName.exiting(*params*);	Logs the exit point of each method, and takes zero or more comma-separated parameters (*params*).	`LOGGER.exiting(LOGPREFIX+"exiting stuff");`

loggerName.logp(Level.*log-level*, *params*)	Allows you to specify the parameters (*params*) to include in the z/OSMF log for the specified log level (*log-level*). This method takes a variable number of comma-separated parameters, and is an alternative to the *loggerName*.*log-level*(*param*) method.	`LOGGER.logp(Level.WARNING, PACKAGE_NAME, MODULE,`
		`methodName, msg, " targetid=", targetid);`
		`LOGGER.logp(Level.SEVERE, PACKAGE_NAME, MODULE,`
		`methodName, "Exception received attempting`
		`to authenticate user", userid, exception);`

loggerName.isLoggable(Level.*log-level*)	Verifies that the specified log level (*log-level*) is currently being logged.	`if(LOGGER.isLoggable(Level.FINEST)){`
		`LOGGER.finest(LOGPREFIX+ "FTP Host : "`
		`+ dojo.byId('<%= HADDataAdaptor.FTPHOSTNAME %>').`
		`value);`
		`}`

loggerName.logMessage("msgID", msgvars);	Logs the message ID as well as the values that were substituted for each parameter. The values are stored as an array.	`var msgvars = ["tkdole", 10, "try again"];` `LOGGER.logMessage("IZUG400E", msgvars);`
loggerName.turnOnTracing();	Enables tracing by setting the log level to FINER.	`LOGGER.turnOnTracing();`
loggerName.turnOffTracing();	Disables tracing, and sets the log level to its initial level.	`LOGGER.turnOffTracing();`
loggerName.turnOnPopup();	Appends messages to a popup window. The default log level is FINEST. You can also append messages to a popup window by setting the setpopup property to *true*. For more details, see "Logging messages to a popup window" on page 717.	`LOGGER.turnOnPopup();`

Table 343. Methods provided for the client side logger (continued)

Method	Description	Sample Code
loggerName.turnOffPopup();	Stops appending messages to a popup window (default), and closes the popup window. You can also stop appending messages to a popup window by setting the setpopup property to *false*. For more details, see "Logging messages to a popup window" on page 717.	`LOGGER.turnOffPopup();`
loggerName.setLogToConsole(true | false);	If set to *true*, the client side logger appends the messages to the browser console, for example, Firebug. If set to *false* (default), the messages are not appended to the browser console.	`LOGGER.setLogToConsole(true);` `LOGGER.setLogToConsole(false);`
loggerName.flush();	Sends the queued messages to the z/OSMF log. You can also use the push property to push queued messages to the z/OSMF log. For more details, see "Pushing messages to the z/OSMF log" on page 717.	`LOGGER.flush();`

Sample z/OSMF client side log data

The message entries in the z/OSMF log have a uniform structure, which is depicted in Figure 357.

```
2009-04-29T22:08:09.609Z|00000031|com.ibm.zoszmf.util.log.servlet.UILoggerServlet|UILoggerServlet::doPost()
FINER: [2009-04-29T22:07:13.938Z] ENTRY MYMODULE MYPACKAGE.jsp init: entry stuff
[tx0000000000002183:zosmfad@localhost (POST) /zosmf/IzuUICommon/UILoggerServlet?preventCache=1240947046138]
2009-04-29T22:08:09.609Z|00000031|com.ibm.zoszmf.util.log.servlet.UILoggerServlet|UILoggerServlet::doPost()
FINER: [2009-04-29T22:07:13.474Z] RETURN MYMODULE MYPACKAGE.jsp init: exiting stuff
[tx0000000000002184:zosmfad@localhost (POST) /zosmf/IzuUICommon/UILoggerServlet?preventCache=1240946722135]
```

Figure 357. Sample z/OSMF client side log data

The first line of a log record contains the following data:

- Date and time the message was added to the log in ISO8601 format, set to UTC timezone. Example: 2009-04-29T22:08:09.609Z.
- Thread ID as an 8 digit hex number. Example: 00000031.
- Class name. Example: com.ibm.zoszmf.util.log.servlet.UILoggerServlet.
- Method name. Example: UILoggerServlet::doPost().

The next line of a log record contains the following data:

- Logging level. Possible logging levels include FINE, FINER, FINEST, INFO, WARNING, and SEVERE.
- Date and time the message occurred in ISO8601 format, set to UTC timezone. Example: [2009-04-29T22:07:13.938Z].
- Indicator of the beginning (ENTRY) or end (RETURN) of a routine if the logging level is FINER.
- Log prefix if a prefix was specified for the logging level.
- Message ID and message text. Message IDs that begin with "IZU" are part of the z/OSMF product.

If the log record includes an exception, the exception class and the message text are logged next followed by the traceback information that is embedded in the exception. If the exception has attached causes, each cause is also logged with "+->" indicating the start of an attached cause.

The final line of a log record contains the following data:

- Transaction ID, which is an internal counter that applies to all actions between a specific set and is clear of a context.
- User ID of the user who was logged into z/OSMF when the message was issued.
- Host name of the system where the user logged into z/OSMF.
- Servlet "verb". Examples include (GET) and (POST).
- URL of the request and query string.

Modifying the default settings for the client side logger

z/OSMF core provides default settings for several properties used to manage the client side logger. This section explains how to modify those default settings.

The default settings are described in the following sections:

- “Displaying the default settings” on page 717
- “Pushing messages to the z/OSMF log” on page 717
- “Logging messages to a popup window” on page 717
- “Redirecting messages that cannot be written to the z/OSMF log” on page 718
- “Setting the interval for flushing the queue” on page 718

Displaying the default settings

To display a list of all the default values for the client side logger properties, use the following HTTP request:

```
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=getprops&loggername={name}
```

where:

- "https://{host}:{port}" specifies the target system address and port.
- "zosmf/IzuUICommon/UILogManager" identifies the client logger interface.
- "OPER=getprops&loggername=name" returns all the default settings for the specified client side logger (*name*).

Pushing messages to the z/OSMF log

The client side logger uses the severity of JavaScript log statements to determine when queued messages are pushed to the z/OSMF log. That is, when the severity of a JavaScript log statement equals or exceeds the push level, the client side logger immediately sends the queued messages to the z/OSMF log. The default push level is INFO. To manage the push level, use the following HTTP requests:

```
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=getLevel&loggername={name}
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=setLevel&level={level}&
    loggername={name}
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=push&loggername={name}
```

where:

- "https://{host}:{port}" specifies the target system address and port.
- "zosmf/IzuUICommon/UILogManager" identifies the client logger interface.
- "OPER=getLevel&loggername={name}" returns the current push level for the specified client side logger (*name*).
- "OPER=setLevel&level={level}&loggername={name}" sets the push level (*level*) for the specified client side logger (*name*), and returns the push level. Valid levels are FINE, FINER, FINEST, INFO, WARNING, and SEVERE.
- "OPER=push&loggername={name}" pushes the queued messages for the specified client side logger (*name*) to the z/OSMF log.

Logging messages to a popup window

By default, the client side logger routes the JavaScript log statements to the z/OSMF log. You can use the *setpopup* property to route the statements to a popup window in the z/OSMF user interface. To indicate where to display the JavaScript log statements, use the following HTTP requests:

```
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=getpopup&loggername={name}
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=setpopup&loggername={name}
    &popup=true|false
```

where:

- "https://{host}:{port}" specifies the target system address and port.
- "zosmf/IzuUICommon/UILogManager" identifies the client logger interface.

- "OPER=getpopup&loggername=name" returns *true* if the specified client side logger (*name*) will append messages to a popup window. Otherwise, *false* is returned.
- "OPER=setpopup&loggername=name&popup=true|false" indicates whether to append messages to a popup window for the specified client side logger (*name*). Set this property to *true* to append messages to a popup window. Otherwise, set this property to *false*.

If you specify *true* for the *setfailnotify* property, messages will also be displayed in a popup window.

Redirecting messages that cannot be written to the z/OSMF log

If an error occurs that prevents the client's messages from being written to the z/OSMF log, you can use the *setfailnotify* property to indicate whether to redirect those messages to a popup window. This failover action allows for the client data to be retained until the error is resolved. By default, this property is set to *false*.

```
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=getfailnotify&
    loggername={name}
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=setfailnotify&
    loggername={name}&failnotify=true|false
```

where:

- "https://{host}:{port}" specifies the target system address and port.
- "zosmf/IzuUICommon/UILogManager" identifies the client logger interface.
- "OPER=getfailnotify&loggername=name" returns *true* if the specified client side logger (*name*) will redirect messages to a popup window when an unexpected error occurs. Otherwise, *false* is returned.
- "OPER=setfailnotify&loggername=name&failnotify=true|false" indicates whether to redirect messages to a popup window for the specified client side logger (*name*). Set this property to *true* to redirect messages to a popup window when an error occurs. Otherwise, set this property to *false*.

Setting the interval for flushing the queue

The client side logger uses the flush interval to determine how often to flush the message queue and send messages to the z/OSMF log. By default, the logger flushes the message queue every 60 seconds. You can use the *setinterval* property to modify this value.

```
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=getinterval&
    loggername={name}
GET https://{host}:{port}/zosmf/IzuUICommon/UILogManager?_OPER=setinterval&
    loggername={name}&interval={n}
```

where:

- "https://{host}:{port}" specifies the target system address and port.
- "zosmf/IzuUICommon/UILogManager" identifies the client logger interface.
- "OPER=getinterval&loggername=name" returns the flush interval for the specified client side logger (*name*).
- "OPER=setinterval&loggername=name&interval=n" sets the flush interval for the specified client side logger (*name*) to the specified number of seconds (*n*). You can specify a minimum of 10 seconds, and a maximum of 3600 seconds.

Retrieving files and resources for your application

z/OSMF provides a file retrieval service that a client application can use to retrieve the files and resources required for the application to display and function properly.

To request files and resources from the z/OSMF file retrieval service, use the following URL format:

```
GET https://{host}:{port}/zosmf/IzuUICommon/externalfiles/{resourcePath}
```

where:

- "https://{host}:{port}" specifies the hostname or IP address and the port of the target system.
- "zosmf/IzuUICommon/externalfiles" identifies the z/OSMF file retrieval service.
- "resourcePath" identifies the file to be retrieved. The path must start with the plug-in context root subdirectory, which is specified in the plug-in's property file. For more details about the plug-in context root, see "Adding your applications to z/OSMF" on page 735.

When you issue this HTTP request, the file retrieval service:

- Retrieves the specified file from the UNIX file system.
- Provides the file to z/OSMF core to be displayed.
- Loads the code for the user interface.

Typically, the file retrieval service is used to provide z/OSMF core with the URL to use to launch the application when a user clicks the corresponding task name in the z/OSMF navigation area. It can also be used to retrieve additional files and resources as requested by your application.

Example

The following example retrieves the file *myapp.js* for the *myapp* application.

```
GET /zosmf/IzuUICommon/externalfiles/myappcontextroot/myapp.js
Host: 1.56.82.158:80
```

Figure 358. Sample request to retrieve a file

The HTTP response body depends on the type of file to be retrieved. For the previous example, the expected response follows:

```
HTTP/1.1 200 OK
Date: Thu, 13 Jan 2014 05:39:28 +0000GMT
Content-Type: application/x-javascript

(file content...)
```

Figure 359. Sample response for a request to retrieve a file

Authoring end user assistance

z/OSMF provides a help system that familiarizes users with the interface, teaches users the concepts required to perform the supported tasks, and helps users troubleshoot errors and transition from one step to another. z/OSMF allows you to add documentation to the help system so you can provide end user assistance that enables users to effectively and easily use your application.

Overview of the z/OSMF help system

The z/OSMF help system, which is integrated into the software product, contains user assistance for each page, window, message, and action supported in the z/OSMF interface. Users can access the help system by clicking the help link, help button, or message ID link provided in the interface. When clicked, context-sensitive help is displayed within the z/OSMF help system framework, which is depicted in Figure 360.

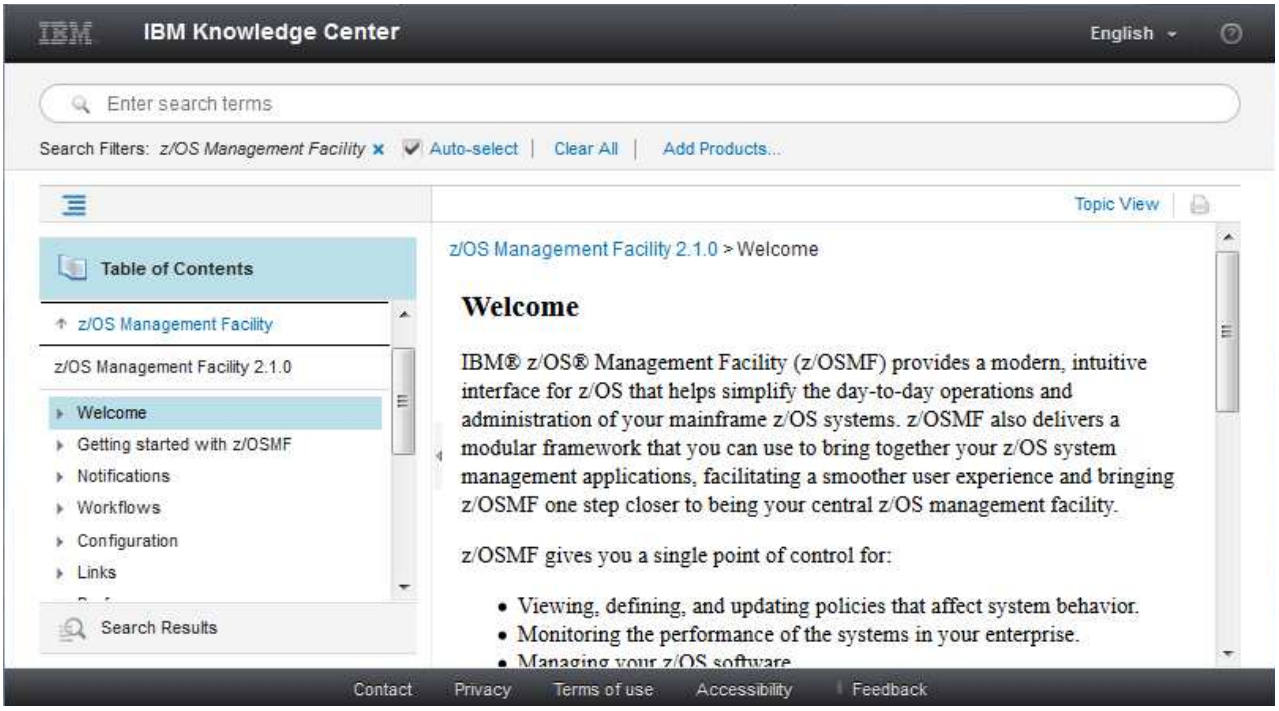

Figure 360. Screen capture of the z/OSMF help system

The framework displays the table of contents and the requested help file. The table of contents mirrors the navigation tree provided in the z/OSMF interface. That is, an application (task) and its help content are included in the same category. For example, the user interface and help content for the Incident Log task are contained in the Problem Determination category.

The help system contains the following additional categories:

- **Getting started with z/OSMF**. This section introduces users to the common features of z/OSMF including using tables and wizards, logging in, and navigating.
- **z/OSMF messages**. This section provides a detailed explanation of each z/OSMF message; describes the reason codes (if any) that are listed in each message; and, suggests actions you can perform to resolve the issue.
- **Tools and techniques for troubleshooting**. This section describes the tools and techniques that are available for troubleshooting problems with z/OSMF.

Format of the files in the z/OSMF help system

The help files in the z/OSMF help system are stored as help plug-ins. A *help plug-in* is a folder that contains the following content:

- **doc.zip file**: Contains the help files, which are coded using the XHTML tagging language.
- **toc.xml file**: Provides the table of contents for the plug-in in XML format.
- **index.xml file**: Defines the index entries for the plug-in in XML format.
- **plugin.xml file**: Describes the plug-in to the z/OSMF help system using the XML tagging language.
- **nl folder**: Contains language sub-folders, identified by the 2-character language code, for each language into which the help plug-ins are translated. For example, if the help files are translated into Japanese, the *nl* folder will contain a *ja* sub-folder that contains the Japanese version of the content.

To create help plug-ins and add links to your help files, complete the tasks described in the sections that follow.

Creating help plug-ins

A help plug-in contains the files required to display your application's help files in the z/OSMF help system.

Procedure

1. To create a panel help plug-in, which is the plug-in that contains the help files for each page, window, and action provided in your application, complete the following steps:
   a. Create a folder for the panel help that has the name *com.company-name.task-name.help.doc*, where *company-name* is your company's name and *task-name* is the task name that will be displayed in the z/OSMF navigation area. For example, *com.ibm.incidentlog.help.doc*.
   b. Store the folder in the UNIX file system, and set 755 permissions for the folder. For all files stored in this folder, set 644 permissions.
   c. Create the help content, and code it using the XHTML tagging language. Then, combine the files into a single compressed folder named *doc.zip*. For more details, see "Developing panel help" on page 722.
   d. Create the table of contents in XML format, and name the file *toc.xml*. For instructions, see "Creating the table of contents" on page 726.
   e. Assign the plug-in to the same category you plan to use for your application in the z/OSMF navigation area. For instructions, see "Categorizing help plug-ins" on page 729.
   f. Use the XML tagging language to create a file named *plugin.xml*, which describes the plug-in to the z/OSMF help system. For instructions, see "Identifying the contents of help plug-ins" on page 732.
   g. Store the *doc.zip* folder, *toc.xml* file, *index.xml* file, and *plugin.xml* file in the folder you created.
   h. In the plug-in folder, create an *nl* sub-folder that contains a folder for each language in which the help content is translated, if any. Store the translated files in the correct language folder, which is identified by a 2-character language code. For a list of language codes, see http://www.loc.gov/standards/iso639-2/php/code_list.php.
2. To create a message help plug-in, which is the plug-in that contains the help files for each message your application issues, complete all of the previous steps with the following exceptions:
   - Store the resulting *doc.zip* folder, *toc.xml* file, *index.xml* file, *plugin.xml* file, and *nl* folder in a folder named *com.company-name.task-name.**message**.help.doc*.
   - Assign the message help plug-in to the z/OSMF messages category.

   For information about creating messages and message help, see "Developing message help" on page 724.

Results

If you created a panel help plug-in named *com.mycompany.mytask.help.doc* that is translated into Spanish and Japanese, the plug-in will have the following structure:

- com.mycompany.mytask.help.doc
  - doc.zip
  - index.xml
  - plugin.xml
  - toc.xml
  - nl
    - es
      - doc.zip
      - index.xml
      - plugin.xml
      - toc.xml
    - ja
      - doc.zip
      - index.xml
      - plugin.xml
      - toc.xml

Developing panel help

A panel-help plug-in contains a set of topics, which are independent units of information that are meaningful when displayed alone. This topic-based structure allows you to create context-sensitive help for each page and window displayed in your application. Topic-based content can contain task, concept, or reference information.

These information types are described in the following sections:

- "Task topics"
- "Concept topics" on page 723
- "Reference topics" on page 724

To provide context-sensitive help for your application, use the XHTML tagging language to create the following topics:

- An introductory topic that provides a brief description of the task and highlights its key features.
- A separate topic for each page or window in the user interface that explains the purpose of the panel and describes the elements on the panel including any fields, columns, actions, or buttons.
- A separate topic for each action that provides step-by-step instructions for performing the action.
- A separate topic for any concept or reference information required for users to effectively operate your application.

Task topics

A task topic uses a series of steps to explain how to accomplish a goal. Task topics provide procedures, typically in step-by-step instructions. Some task topics might list choices, as bulleted points rather than steps, or they might describe a single action rather than a sequence of steps. Tasks topics also provide information about the context (where to perform a task and when), the rationale (why perform the task), prerequisites, and examples.

Supertasks (high-level tasks) are the starting points for most users. Steps in a supertask often link to sub-tasks. Each sub-task is documented as a separate task topic. These tasks can be part of one supertask or many. For example, in database programming, opening a database connection is a task that can be reused in several high-level programming tasks.

A task topic documents what users need to know to successfully complete their work. As you write task information, note what concepts need to be documented, and to what depth, if users are to complete the tasks successfully. Consider the probable skills and experience of users, and write with those characteristics in mind. For example, if users need to configure TCP/IP (a task) and might not know about routers, create and link to a concept topic on routers.

Remember these main points when you write task topics:

- Document the steps that users follow to accomplish their goals.
- Use a verb phrase (gerund) as the heading for a task topic.
- Use an opening paragraph to provide context for and introduce the task.
- If the task has a prerequisite, provide that information or provide a link to that prerequisite before the list of steps.
- Use a numbered list for the steps that users must follow to complete the task.
- Write the steps as brief imperative sentences.
- When the user's context changes, introduce the step with a phrase that establishes that new context (for example, On the Configuration page, ...).
- Write one step for each significant user action.
- If a task has more than nine steps, try to divide it into two or more separate tasks.
- If the task is part of a sequence of tasks, provide a link to the next task.

Concept topics

A concept topic describes a system, solution, product, tool, feature, or background information that users need to complete a task. Users typically read concept material before tackling some large project or starting to use a product or tool. In contrast, users need task or reference topics when they perform a task.

A concept topic describes the scope of the topic and clearly defines what the topic is about. Use minimalist writing techniques to create content that users can quickly understand. Beware of turning a concept topic into a white paper by explaining the whole design philosophy of a product or component. Every concept topic performs at least one of the following functions:

- Introduces a solution, process, product, tool, or feature.
- Provides background information and explains issues that users must know before working with a system or component, or before starting a task.
- Describes the benefits of using one approach rather than another, or provides information about when one particular choice or tool is more appropriate than another.
- Describes how one feature, tool, or product is related to others, and how they work together or do not work together.
- Describes any restrictions that limit the circumstances in which a tool can be used successfully.
- Expands the significance of an important term beyond the scope of a glossary definition.
- Explains how and why some behavior changes as time passes or work progresses.
- Helps users form a mental picture that builds on the experience and knowledge that they are already likely to have.

Remember these main points when you write concept topics:

- Document the background knowledge that users need to successfully use the system, process, product, tool, or feature.
- Use a noun or noun phrase as the heading.
- Use paragraphs.
- If the topic is longer than two screens of information, use subheadings to break it into sections.
- When introducing a new term, begin with a definition; then, expand that definition.
- Add graphics when they simplify the explanation, for example, to show a process or the relationship among concepts.
- Provide examples to bridge from unknown knowledge to known.
- Address only one complete idea.
- Keep the concept topic short and concise, but describe the concept completely.

Reference topics

Reference topics provide quick access to information that users are likely to need as they complete tasks. Use reference topics to document the purpose of an element, and any restrictions (such as case sensitivity), required authorizations, or anything else that might limit the use of an element.

There are several types of elements for which you can provide reference information. A few examples follow:

- APIs
- Commands
- Language and programming elements
- Class descriptions
- Keyboard shortcuts
- Protocols
- Schemas
- Settings
- Symbols
- Templates
- Column, field, and action descriptions

Remember these main points when you write reference topics:

- Use tables and lists to make reference information easy to scan.
- Use a noun or noun phrase as the title for a reference topic.
- For a particular category of information, such as API documentation, use a consistent format so users can find information quickly.
- Be brief, but write in full sentences.
- Do not go into long explanations; assume that readers understand the basic technology.
- Make the topic as long as it takes to explain the subject.
- Provide links to closely related reference topics, and in some cases, to related concept and task topics.

Developing message help

A message is any communication that is passed from the application to a user or to another application. z/OSMF uses messages to inform users of important events, such as state changes and errors that require resolution.

For the messages your application issues, use the XHTML tagging language to create a separate help file for each message. Include the following attributes for each message:

- **message ID**. Specifies a unique alphanumeric identifier that provides a quick means to distinguish one message from others. Use the message ID as the filename and topic title.

  Vendor message IDs can start with the letters J-Z; the letters A-I are reserved for IBM. The last character indicates the severity of the message, which can have one of the following values:
  - ***I* for information**. Describes information or status for normal conditions and operations.
  - ***W* for warning**. Alerts users to a condition that might cause problems in the future. When a warning message is displayed, users can generally continue with their tasks, but those tasks might not complete in a way that is expected.
  - ***E* for error**. Alerts users to a problem that already occurred. Users or systems cannot continue their tasks.

  For example, *YYZR134I*.
- **message text**. Briefly describes the problem or situation from the user's perspective. Messages can have variables (or arguments), which are typically numbers or placeholders that are used in a message in place of a specific file name, command, component, or other object. In the message text, ensure that you:
  - Focus on the problem, not the error.
  - Describe the problem briefly, use full sentences, and ensure that the information is accurate.
  - Avoid wording that seems to blame the user.
  - Do not concatenate multiple messages to create a single message.
  - Make variables meaningful and unique.
  - Replace variables only with proper nouns.
  - Use double quotation marks around variables only when necessary.
- **explanation**. Expands the message text and provides more detail. In the explanation, ensure that you:
  - Explain why the message was issued.
  - For error and warning messages, describe the cause of the problem (when and where the error occurred), explain the consequences of the error, and provide information to help users avoid the problem in the future if possible.
  - Do not repeat the message text in the explanation section.
  - Avoid using codes to build messages and resolve problems in error handling. However, if your application uses codes, describe the codes consistently in your messages and include corrective action so users do not need to look elsewhere for an explanation of the code.
- **userResponse** and **sysprogResponse**. Describes what the user (userResponse) or system programmer and administrator (sysprogResponse) must do to proceed, to recover from the error, or to prevent a problem. If no response is required, enter *No action is required.* For warning and error messages, a response must be provided for the system programmer, the user, or both.

  In the response, ensure that you:
  - Use active voice when possible.
  - Provide complete and specific instructions to resolve the problem.
  - Link to other information if necessary.
  - Categorize the actions.
  - Do not leave the response section empty.
  - Ensure that wrapped, associated, and stacked messages are consistently presented.

Checklist for writing effective messages:

User perceptions of a software product are strongly influenced by how well messages convey relevant information and help the user solve a problem. Use the following checklist to ensure that messages are clear, accurate, complete, and helpful.

- Write accurate messages:

- __ Ensure that all facts in the message are accurate.
- __ Avoid product names and versions because they change over time.

- Avoid telling users to call a support organization or system administrator:
  - __ Check whether there are alternative solutions.
  - __ Ask users to check documentation, Web sites, and so on to find solutions.
- Do not blame the user:
  - __ Write messages so that they don't appear to blame the user, even if it the cause is a user error.
  - __ Avoid "doomsday" phrases such as *catastrophic failure* or *fatal error*.
- Use clear language:
  - __ Check for ungrammatical or incorrectly punctuated sentences.
  - __ Avoid garbled, long (over 25 words), or convoluted sentences.
  - __ Check for unnecessary passive voice.
  - __ Avoid abstract language, general language, and jargon.
  - __ Use a consistent style and word usage.
  - __ Always use full sentences with correct punctuation.
  - __ Use nouns after command and API names, for example, the BaseException class. (class is the noun.)
  - __ Do not leave out articles such as *a*, *an*, or *the*.
- Provide complete information:
  - __ Ensure that each message is needed in the system. Don't display a message when the code should handle the issue.
  - __ Provide all the instructions for resolving a problem in one message whenever possible.
  - __ If the user actions are too long or topics for the appropriate corrective actions exist elsewhere, provide a link from the message to other appropriate topics.
  - __ Add explanation, user response, and system programmer response sections to all messages.
  - __ Do not give simplistic user actions such as "See the log files" without more guidance. For example, if you ask users to see log files, tell them where to look in the directory structure to find the log and what to look for in the log.
  - __ Provide examples for commands, APIs, or other code unless the code is extensive or complicated.
  - __ Do not expose unnecessary information such as documenting system actions that do not affect the user, providing information that is too detailed for the target audience, and describing internal workings that the user has no control over. This information should be logged only in a trace for support.

Creating the table of contents

A table of contents is required to integrate each help plug-in into the z/OSMF help system. Providing a table of contents improves the navigability of help plug-ins and ultimately makes it easier for users to find relevant help information. This topic describes the structure of the table of contents for panel and message help plug-ins.

Panel help plug-ins

To create a table of contents for panel help plug-ins, create an XML file that has the following structure:

```
<toc label="task-name" link_to="category" topic="path-to-parent-topic">
 <topic label="task-name" href="path-to-parent-topic">
   <topic label="topic-name" href="path-to-topic"/>
   <topic label="topic-name" href="path-to-topic">
     <topic label="topic-name" href="path-to-topic"/>
     <topic label="topic-name" href="path-to-topic"/>
     <topic label="topic-name" href="path-to-topic"/>
   </topic>
   <topic label="topic-name" href="path-to-topic"/>
 </topic>
</toc>
```

Figure 361. Table of contents template for panel help plug-ins

where:

task-name
Name that will be displayed in the z/OSMF navigation area for your application followed by the word *task*.

category
Path to the category that will contain the help for the task. For more details, see "Categorizing help plug-ins" on page 729.

path-to-parent-topic
Path to and name of the file that introduces the task and its key features. The introductory topic must be the container (parent) for all the other help topics in the plug-in.

topic-name
Name of the topic, which is the label that will be displayed in the table of contents in the z/OSMF help system.

path-to-topic
Path to and name of the file that contains the help content.

Figure 362 provides a sample table of contents for the System Status task.

```
<toc label="System Status task"
 link_to="../com.ibm.zosmfcore.performance.help.doc/izuG00hpPerformance.xml#sysplex"
 topic="izuR00hpSysplexStatusTask.html">
  <topic label="System Status task" href="izuR00hpSysplexStatusTask.html">
   <topic label="Managing system resources" href="izuR00hpSysplexStatusPanel.html">
    <topic label="Adding resource entries" href="izuR00hpAddSysEntry.html">
     <topic label="Add and Modify Entry pages" href="izuR00hpAddModSysPanel.html"/>
    </topic>
    <topic label="Modifying resource entries" href="izuR00hpModSysEntry.html"/>
    <topic label="Removing resource entries" href="izuR00hpRemSysEntry.html"/>
   </topic>
  </topic>
</toc>
```

Figure 362. Sample table of contents for the System Status task

When the panel help plug-in is displayed within the z/OSMF help system, it is listed under the specified category and each help topic is nested under the parent topic, as depicted in Figure 363 on page 728.

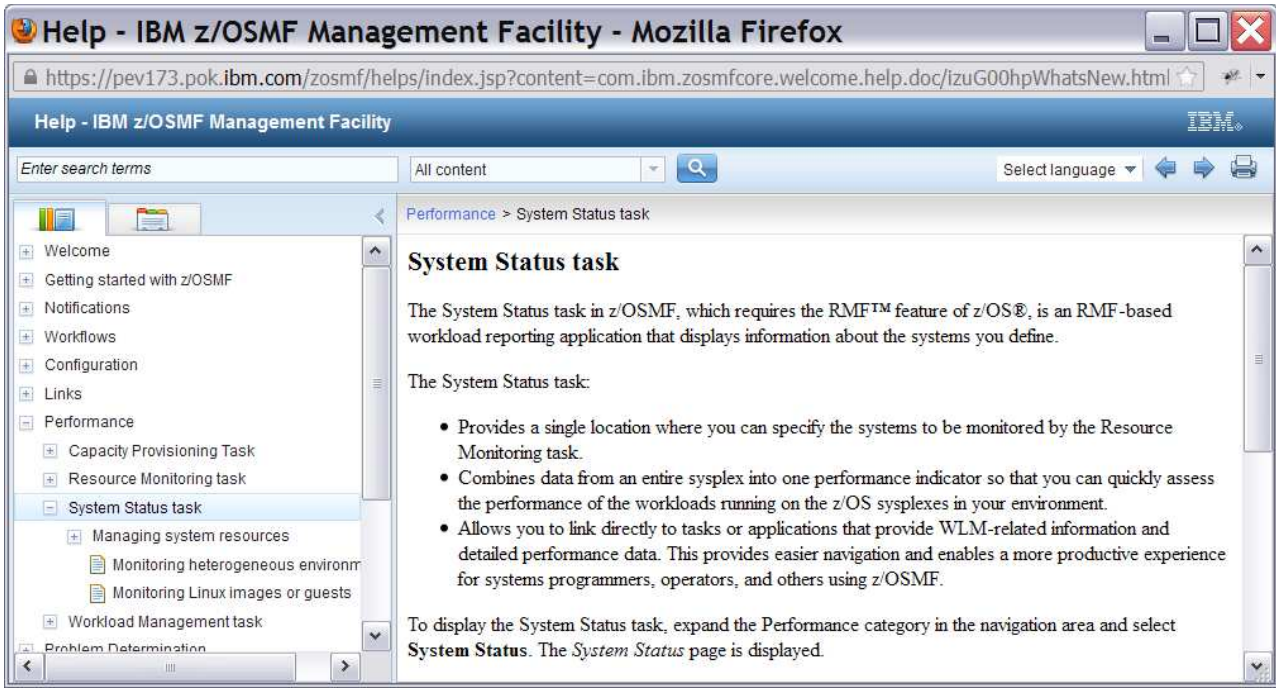

Figure 363. Sample table of contents for panel help plug-ins

Message help plug-ins

To create a table of contents for message help plug-ins, create an XML file that has the following structure:

```
<toc label="message-ID-range" link_to="category" topic="path-to-parent-topic">
 <topic label="message-ID-range" href="path-to-parent-topic">
   <topic label="message-ID" href="path-to-topic">
   <topic label="message-ID" href="path-to-topic"/>
   <topic label="message-ID" href="path-to-topic"/>
   <topic label="message-ID" href="path-to-topic"/>
 </topic>
</toc>
```

Figure 364. Table of contents template for message help plug-ins

where:

message-ID-range
Range of message IDs included in the message help plug-in.

category
Path to the z/OSMF messages category, which will contain the message help for the task. For more details, see "Categorizing help plug-ins" on page 729.

path-to-parent-topic
Path to and name of the file that is the container (parent) for all of the message help topics. The parent topic should state the following: *This topic describes the z/OSMF messages that have a message ID between message-ID-range*.

topic-name
Name of the topic, which is the ID of the message.

path-to-topic
Path to and name of the file that contains user assistance for the message.

Figure 365 provides a sample table of contents for messages issued by z/OSMF core.

```
<toc label="IZUG0400-IZUG9999"
 link_to="../com.ibm.zosmfmessages.help.doc/izuAllhpzOSMFMessages.xml#core_messages"
 topic="IZUG0400-IZUG9999.html">
  <topic label="IZUG0400-IZUG9999" href="IZUG0400-IZUG9999.html">
   <topic label="IZUG400I" href="IZUG400I.html"/>
   <topic label="IZUG401E" href="IZUG401E.html"/>
   <topic label="IZUG401W" href="IZUG401W.html"/>
   <topic label="IZUG402W" href="IZUG402W.html"/>
   <topic label="IZUG403E" href="IZUG403E.html"/>
   <topic label="IZUG404I" href="IZUG404I.html"/>
  </topic>
</toc>
```

Figure 365. Sample table of contents for the messages issued by z/OSMF core

When the message help plug-in is displayed within the z/OSMF help system, it is listed under the z/OSMF messages category and each message topic is nested under the parent topic, as depicted in Figure 366.

Figure 366. Sample table of contents for message help plug-ins

Categorizing help plug-ins

In the z/OSMF help system, a task and its panel help plug-in are included in the same category, and the task's message help plug-in is listed under the z/OSMF messages category, where all z/OSMF messages are listed. This structure ensures consistency across z/OSMF tasks, making it easier for users to find relevant content.

To categorize the panel and message help plug-ins, on the `<toc>` element in the *toc.xml* file, specify the appropriate URL for the *link_to* attribute. Table 344 on page 731 lists the URL to use for each z/OSMF

category. For example, to add the message help plug-in to the z/OSMF messages category, type the following value: *link_to="../com.ibm.zosmfmessages.help.doc/izuAllhpzOSMFMessages.xml#messages"*.

Table 344. URL to use for each z/OSMF category

Category	URL
Commands and Logs	../com.ibm.zosmfcore.commandlog.help.doc/izuG00hpCommandLogs.xml#commands_and_logs
Configuration	../com.ibm.zosmfcore.configuration.help.doc/izuG00hpzOSMFConfiguration.xml#configuration
Jobs and Resources	../com.ibm.zosmfcore.jobresources.help.doc/izuG00hpJobResources.xml#jobs_and_resources
Links	../com.ibm.zosmfcore.linksuser.help.doc/izuG00hpLinksUser.xml#links
Performance	../com.ibm.zosmfcore.performance.help.doc/izuG00hpPerformance.xml#performance
Problem Determination	../com.ibm.zosmfcore.problemdetermination.help.doc/izuAllhpzOSMFProbDet.xml#problem_determination
Software	../com.ibm.zosmfcore.software.help.doc/izuG00hpSoftware.xml#software
z/OS Classic Interfaces	../com.ibm.zosmfcore.classicinterfaces.help.doc/izuG00hpClassicInterfaces.xml#classic_interfaces
z/OSMF Administration	../com.ibm.zosmfcore.administration.help.doc/izuG00hpzOSMFAdministration.xml#administration
z/OSMF messages	../com.ibm.zosmfmessages.help.doc/izuAllhpzOSMFMessages.xml#messages
z/OSMF Settings	../com.ibm.zosmfcore.settings.help.doc/izuG00hpSettings.xml#settings

Identifying the contents of help plug-ins

Every plug-in requires a file called *plugin.xml* to identify the plug-in contents to the z/OSMF help system.

About this task

This file includes the following items:

- The table of contents file that contributes to the navigation for the help plug-in.
- The name, ID, and the version of the help plug-in.
- The index file for building the Index view.

Procedure

To create the *plugin.xml* file for your help plug-ins, follow these steps:

1. Create a new Extensible Markup Language (XML) file called *plugin.xml*.
2. Make the following XML declaration the first two lines of the file:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <?eclipse version="3.0"?>
   ```

3. Add the root <plugin> element and specify the following attributes for the help plug-in:

   name For the panel help plug-in, the name that will be displayed in the z/OSMF navigation area for your application followed by the word *task*.

   For the message help plug-in, the range of message IDs included in the plug-in.

   id Name of the folder that contains the panel or message help contents.

   version
   Version of the help plug-in.

   provider-name
   Name of your company.

   For example, the plug-in identification metadata for the Incident Log task can be as follows:

   ```
   <plugin name = "Incident Log task"
           id = "com.ibm.zosmfincidentlog.help.doc"
           version ="2.1"
           provider-name = "IBM">
   ```

4. Within the context of the <plugin> element, create an <extension> element with a *point* attribute value of *org.eclipse.help.toc*. Then, within the context of the <extension> element, create a <toc> element to declare the table of contents file.

   ```
   <extension point="org.eclipse.help.toc">
      <toc file="toc.xml" primary="true" />
   </extension>
   ```

5. If you indexed your content, within the context of the <plugin> element, create an <extension> element with a *point* attribute value of *org.eclipse.help.index*. Then, within the context of the <extension> element, create an <index> element to declare the index file.

   ```
   <extension point="org.eclipse.help.index">
      <index file="index.xml" />
   </extension>
   ```

6. Close the <plugin> element.

   ```
   </plugin>
   ```

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>

<plugin name = "Incident Log task"
        id = "com.ibm.zosmfincidentlog.help.doc"
```

```
        version ="2.1"
        vendor-name = "IBM">

  <extension point="org.eclipse.help.toc">
    <toc file="toc.xml" primary="true" />
  </extension>

  <extension point="org.eclipse.help.index">
    <index file="index.xml" />
  </extension>

</plugin>
```

Adding links to help plug-ins

z/OSMF tasks provide a help link, help button, or message ID link on every page, window, or message that is provided in the user interface. When users click the corresponding widget, user assistance is displayed for the current context.

Procedure

To provide context-sensitive help for your applications, complete the steps that follow:

1. Create a help topic for each page, window, and message that can be displayed in your applications. For instructions, see "Creating help plug-ins" on page 721.
2. Add a help link, help button, message ID link, or another widget to each page, window, and message.
3. For each widget, specify the URL for the related help topic. The URL must have the following format:

   `https://<host>:<port>/<context-root>/helps/SSB2H8_2.1.0/<help-plugin-name>/<help-topic>`

   where:

   host Hostname or IP address of the system where z/OSMF is installed.

   port Secure application port for the z/OSMF configuration. If you specified a secure port for SSL encrypted traffic during the configuration process (through the IZUPRMxx parmlib member keyword HTTP_SSL_PORT), specify the same port in the URL. If you omit the port, it is assumed that you are using port 443, the default.

   context-root
   Context root of the z/OSMF application. By default, the context root is `zosmf`.

   help-plugin-name
   Name of the help plug-in that contains the help topic to which you want to link.

   help-topic
   Filename of the help topic to which you want to link.

Example

Figure 367 on page 734 provides sample code for linking a button to a specific help topic in the z/OSMF help system.

```
<html>
 <head>
  <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
  <script type="text/javascript">
   function open_help()
   {
    window.open(
    "https://abc.com:81/zosmf/helps/SSB2H8_2.1.0/com.ibm.task.help.doc/about.html","help"
    );
   }
  </script>
 </head>
 <body>
  <input value="Help" onclick="open_help()" type="button">
 </body>
</html>
```

Figure 367. Sample code for linking to a help topic

Adding your applications to z/OSMF

To add your application to z/OSMF, create a property file that defines the parameters required for z/OSMF to configure your application, and use the z/OSMF Import Manager task to import the property file.

Before you begin

- Develop a web-based application and the supporting documentation for the functions you want to add to z/OSMF. For instructions, see "Developing web-based applications" on page 684 and "Authoring end user assistance" on page 720.
- Identify and resolve any security vulnerabilities in your application. For more details, see "Verifying the security of applications" on page 741.

About this task

A property file is a flat file, such as a text file, that contains a set of attributes for one or more instances of an object. The attributes are specified as name and value pairs, and must be enumerated for each instance of the object. z/OSMF supports property files that are encoded using the platform default encoding, which by default is EBCDIC 1047, or ASCII.

When you use a property file to import a plug-in into z/OSMF, z/OSMF core:

- Validates the structure of the property file, and verifies that a value is provided for the required properties.
- Creates symbolic links to the client-side code for your application so that any updates to that code is reflected in z/OSMF.
- Stores the symbolic links in the plug-in context root subdirectory, which is specified in the plug-in property file.
- Adds a task defined in the property file to the z/OSMF navigation area if a URL is specified for its taskNavigationURL property.

Procedure

To use a property file to add a new plug-in to z/OSMF, complete the steps that follow:

1. Create a property file in the z/OS UNIX System Services (z/OS UNIX) file system, and set 644 as its permissions. For example, you can create a text file called *plugin.properties*.
2. Type *importType=plugin* at the beginning of the file. Doing so informs z/OSMF that the property file contains a plug-in definition. You cannot define other types of objects, such as event types or handlers, in a property file for a plug-in.
3. Define only one plug-in to be added to z/OSMF. To do so, specify the following properties for the new plug-in:

```
izu.externalapp.file.version=plugin-version
izu.externalapp.local.context.root=plugin-context-root
izu.externalapp.code.root=plugin-code-root

pluginId=plugin-Id
pluginDefaultName=plugin-name
pluginDescription=plugin-description
aboutPanelPath=about-panel-path

izu.externalapp.help.root=help-root
izu.externalapp.helpdoc=help-plugin-name
```

plugin-version
Specify the version of the plug-in.

plugin-context-root
Specify the name of the symbolic link directory under the `USERDIR/data/externalapps` directory to use for your applications. The USERDIR setting, which identifies the mount point of the z/OSMF user file system, is set when your installation configures z/OSMF. By default, the mount point is /var/zosmf/.

If you specify `myapp` as the plug-in context root, a symbolic link named `myapp` will be created in the path of `USERDIR/data/externalapps/myapp`, and it will point to the source code path `/usr/lpp/myapp/ui`.

z/OSMF uses the context root to build the navigation URL and the bundle URL.

The plug-in context root is required, and the directory name must comply with the z/OS UNIX naming guidelines.

plugin-code-root
Specify the path to the z/OS UNIX directory that contains the source code for the tasks included in the plug-in. If the plug-in property file and the source code directory reside in the same directory, you can specify a relative path, for example, `ui`. Otherwise, you must specify the absolute path, for example, `/usr/lpp/myapp/ui`. The plug-in code root is required.

plugin-ID
Specify a unique identifier for the plug-in. The ID is required, must be unique, and can contain a maximum of 64 characters.

plugin-name
Specify the name of the plug-in. The plug-in name is required, and can contain a maximum of 64 characters.

plugin-description
Provide a description of the plug-in. The description is required, and can contain a maximum of 256 characters.

about-panel-path
Specify the absolute path of the flat file that contains the plug-in version information to display on the z/OSMF About page. For example, `/zosmf/IzuUICommon/externalfiles/myapp/myappVersion.txt`. The path is required.

Tip: If the file resides in the same directory as the plug-in property file, you can specify the filename with no path information. For example, `myappVersion.txt`.

help-root
Specify the path to the z/OS UNIX directory that contains the help files for the tasks included in the plug-in. If the plug-in property file and the help root directory reside in the same directory, you can specify a relative path, for example, `helps`. Otherwise, you must specify the absolute path, for example, `/usr/lpp/myapp/helps`. The plug-in help root is optional.

help-plugin-name
Specify the name of the directory that contains the help files for the tasks in your application. For example, `com.ibm.zosmfmyapp.task1.help.doc`. The directory name is required only if you are including help files in your plug-in. The directory name must comply with the z/OS UNIX naming guidelines.

Each help plug-in must be enumerated and must be listed in numeric order because z/OSMF expects the first help plug-in to be enumerated as *izu.externalapp.help.root.1*, the second as *izu.externalapp.help.root.2*, and so on. The first time the enumeration does not match the position of the help plug-in in the file, z/OSMF stops looking for help plug-ins.

4. Include one to 32 task definitions in the plug-in property file. Each task definition must be enumerated and must be listed in numeric order because z/OSMF expects the first task definition to

be enumerated as one, the second definition as two, and so on. The first time the enumeration does not match the position of the task definition in the file, z/OSMF stops reading the file. That is, the remaining task definitions will not processed.

The following attributes are supported for tasks:

```
taskId=task-ID
taskVersion=task-version
taskCategoryId=category-ID
taskDispName=task-name
taskDispDesc=task-description
taskSAFResourceName=task-SAF-resource-name
taskMultiSysplexScope=task-multi-sysplex-scope
taskHandlerEligible=task-handler-eligible
taskAuthenticatedGuestEligible=task-authenticated-guest-eligible
taskNavigationURL=task-navigation-URL
taskBundleUrl=bundle-URL
taskBundleFileName=bundle-file-name
taskMinZOS=minimum-z/OS-level
taskMinZOSMF=minimum-z/OSMF-level
```

task-ID
Specify a unique identifier for the task. The ID is required, must be unique, and can contain a maximum of 64 characters.

task-version
Specify the version of the task.

category-ID
Specify where the task is to be displayed in the z/OSMF navigation area. You can assign the task to any valid category, or you can opt not to categorize the task. If assigned to a category, the task is sorted alphabetically with the other tasks and links in that category. Otherwise, the task is placed after the Welcome task in the navigation area, sorted alphabetically with any other uncategorized tasks and links.

To indicate the placement of the task, specify one of the following values:

- **1** - z/OSMF Administration
- **2** - Problem Determination
- **3** - Links
- **4** - Configuration
- **5** - Software
- **7** - z/OS Classic Interfaces
- **9** - Performance
- **10** - z/OSMF Settings
- **11** - Uncategorized
- **12** - Commands and Logs
- **13** - Jobs and Resources

The category ID is required.

task-name
Specify the task name to be displayed in the z/OSMF navigation area. The task name is required, and can contain a maximum of 30 characters.

task-description
Provide a description of the task. The description is required, and can contain a maximum of 200 characters.

task-SAF-resource-name

Specify a unique SAF resource name to be used for managing user authorizations to the task. The resource name must start with ZOSMF, and must conform to the following structure to ensure uniqueness:

```
ZOSMF.<vendor>_<plugin-ID>.<task-ID>.<task-name>
```

where:

vendor
Name of your company.

plugin-ID
Unique identifier you assigned to the plug-in.

task-ID
Unique identifier you assigned to the task.

task-name
Name you assigned to the task.

For example, ZOSMF.IBM_TESTPLUGIN.COMMANDS.Commands.

If the SAF resource name does not begin with ZOSMF, z/OSMF will add prefix ZOSMF.IMPORT to the SAF resource name.

The resource name is required, and can contain up to 231 alphanumeric characters (A-Z a-z 0-9) and the following special characters: underscore (_), dash (-), period (.). The use of a period in a resource name is treated as a qualifier. As such, the first character after a period must be A-Z or a-z.

task-multi-sysplex-scope

z/OSMF provides a multi-sysplex capability, which allows you to manage multiple z/OS sysplexes from a single z/OSMF instance. To do so, a z/OSMF instance must be running in each sysplex to be managed.

Set the *task-multi-sysplex-scope* property to *true* to indicate that this task can be used to manage or display data for multiple z/OS sysplexes. Otherwise, omit this property or set it to *false*.

task-handler-eligible

z/OSMF provides the application linking capability, which allows you to create context-sensitive launch points between tasks or applications. The task or application that initiates the launch request is referred to as the event requestor, and the task or application that processes the request and displays the appropriate context is referred to as the event handler.

Set the *task-handler-eligible* property to *true* to indicate that the task is eligible to be an event handler in the application linking process. Doing so allows users to define the task as a handler for one or more event types. To disallow the task to participate in the application linking process as an event handler, omit this property or set it to *false*.

task-authenticated-guest-eligible

Set this property to *true* to extend task authorization to users who are logged into z/OSMF, but are not defined to a z/OSMF SAF security group. Otherwise, omit this property or set it to false. Extending task authorization to users who are not logged into z/OSMF is not supported.

task-navigation-URL

Specify the relative or absolute path of the home page for the task. For example, if the home page is named index.html and resides in the myapp directory (context root), you can specify index.html or /zosmf/IzuUICommon/externalfiles/myapp/index.html.

If the plug-in contains more than one task, append the following to the path: ?task=contextRoot.taskName, where *contextRoot* and *taskName* are the values you specified for

the `izu.externalapp.local.context.root` property and the `taskDispName` property. For example, `/zosmf/IzuUICommon/externalfiles/myapp/index.html?myapp.settings.task`.

The path can contain a maximum of 4000 characters, including alphanumeric characters (A-Z a-z 0-9), blanks, mathematical symbols (+ - = | ~ ( ) { } \), punctuation marks (? , . ! ; : ' " / [ ]), and the following special characters: %, $, #, @, ^, *, and _. Any leading or trailing white space is ignored.

The navigation URL is required if the bundle URL is specified. Otherwise, omit the navigation URL.

bundle-URL
Specify the relative or absolute path of the language resource bundle for the task. For example, if the bundle file resides in the `/myapp/js` directory, you can specify `/js` or `/zosmf/IzuUICommon/externalfiles/myapp/js`.

The path can contain a maximum of 256 characters, including alphanumeric characters (A-Z a-z 0-9), blanks, mathematical symbols (+ - = | ~ ( ) { } \), punctuation marks (? , . ! ; : ' " / [ ]), and the following special characters: %, $, #, @, ^, *, and _. Any leading or trailing white space is ignored.

The bundle URL is required if the navigation URL is specified. Otherwise, omit the bundle URL.

bundle-file-name
Specify the name of the language resource bundle file. The file name can contain a maximum of 256 characters. For example, `bundle.js`. The file name is required if the bundle URL is specified.

minimum-z/OS-level
Specify the minimum z/OS operating system level that the task requires. You can specify one of the following values:

- **04.24.00**: Indicates that the minimum z/OS level is V2R1.
- **03.23.00**: Indicates that the minimum z/OS level is V1R13.

minimum-z/OSMF-level
Specify the minimum z/OSMF level that the task requires. You can specify one of the following values:

- **04.24.00**: Indicates that the minimum z/OSMF level is V2R1.
- **03.23.00**: Indicates that the minimum z/OSMF level is V1R13.

Figure 368 on page 740 provides a sample property file that defines the *myapp* plug-in, which contains a Commands task and a Settings task.

```
importType=plugin

izu.externalapp.file.version=1.0.0
izu.externalapp.local.context.root=myapp
izu.externalapp.code.root=ui
pluginId=com.ibm.zoszmf.myapp
pluginDefaultName=myapp
pluginDescription=Operate myapp.
aboutPanelPath=/zosmf/IzuUICommon/externalfiles/myapp/myappVersion.txt

izu.externalapp.help.root=helps
izu.externalapp.helpdoc.1=com.ibm.zosmfmyapp.commands.help.doc
izu.externalapp.helpdoc.2=com.ibm.zosmfmyapp.settings.help.doc

taskId1=COMMANDS
taskVersion1=1.0
taskCategoryId1=12
taskDispName1=Commands
taskDispDesc1=The Commands task lets you enter z/OS commands.
taskSAFResourceName1=ZOSMF.IBM_COMMANDS.COMMANDS.Commands
taskMultiSysplexScope1=true
taskHandlerEligible1=true
taskAuthenticatedGuestEligible1=true
taskNavigationURL1=/zosmf/IzuUICommon/externalfiles/myapp/index.html?task=myapp.commands
taskBundleUrl1=/zosmf/IzuUICommon/externalfiles/myapp/js/nls
taskBundleFileName1=bundle.js
taskMinZOS1=04.24.00
taskMinZOSMF1=04.24.00

taskId2=SETTINGS
taskVersion2=1.0
taskCategoryId2=10
taskDispName2=Settings
taskDispDesc2=The Settings task allows you to define task-specific settings.
taskSAFResourceName2=ZOSMF.IBM_SETTINGS.SETTINGS.Settings
taskHandlerEligible2=true
taskAuthenticatedGuestEligible2=true
taskNavigationURL2=/zosmf/IzuUICommon/externalfiles/myapp/index.html?task=myapp.settings
taskBundleUrl2=/zosmf/IzuUICommon/externalfiles/myapp/js/nls
taskBundleFileName2=bundle.js
taskMinZOS2=04.24.00
taskMinZOSMF2=04.24.00
```

Figure 368. Sample plug-in property file

Tip: To remove a plug-in and all of its tasks from z/OSMF, type *deletePlugin=true* at the end of the property file. Otherwise, omit this property.

5. Save the property file.
6. Import the property file. To do so, complete the following steps:
   a. In the z/OSMF navigation area, expand the z/OSMF Administration category and select **Import Manager**.
   b. On the **Import** tab on the Import Manager page, specify the full path and name of the property file you created, and click **Import**.

Results

A message indicating whether the plug-in was added is displayed. If the plug-in was added, the tasks where you provided a URL for the taskNavigationURL property are listed in the z/OSMF navigation area under the specified category.

If z/OSMF finds any errors in the property file, neither the plug-in nor any of its tasks will be added to z/OSMF. In which case, you must resolve the errors and import the property file again.

What to do next

If the server-side code for your plug-in does not reside on the z/OSMF server, you must associate an application server with each task included in the plug-in. Otherwise, the plug-in will not work correctly. For instructions, see the topic about associating application servers with imported tasks in the z/OSMF help system.

Set up security for your plug-in. After which, you must refresh the security management product on your system and restart the z/OSMF server to have your changes take affect. For more details, see the section about securing your applications in "Securing your applications."

Securing your applications

To secure your applications, identify and resolve any security vulnerabilities, and work with your security administrator to grant users access to your applications. When the required security controls are established on your system, a user can begin using z/OSMF to perform system management tasks.

Verifying the security of applications

Before importing applications into z/OSMF, ensure that the vendor or developer who supplied the application adhered to security best practices for Web applications. If the software installed is not secure, it is possible to expose your system or company to security issues.

Controlling access to applications

After importing your plug-in into z/OSMF, work with your security administrator to authorize users to your applications. z/OSMF security is based on the following concepts:

user authentication
When a user attempts to log in to z/OSMF through a web browser, the user's credentials are verified by the z/OS host system through the SAF interface or a security management product (for example, RACF). This processing ensures that the user ID is known to the z/OS system, and the password is valid.

user authorization
Access to your application is controlled through SAF resource profile <safPrefix>.<taskSAFResourceName>, where <safPrefix> is configured in z/OSMF and is by default IZUDFLT and <taskSAFResourceName> is the SAF resource name you specified for the task in the plug-in property file. The SAF resource profile is defined in the ZMFAPLA class.

If your installation is using RACF and you want to assign administrators CONTROL access and users READ access to your application, you can create a profile like the following:

```
RDEFINE ZMFAPLA +
(IZUDFLT.ZOSMF.IBM_COMMANDS.COMMANDS.Commands) UACC(NONE)
PERMIT +
IZUDFLT.ZOSMF.IBM_COMMANDS.COMMANDS.Commands +
CLASS(ZMFAPLA) ID(IZUADMIN) ACCESS(CONTROL)
PERMIT +
IZUDFLT.ZOSMF.IBM_COMMANDS.COMMANDS.Commands +
CLASS(ZMFAPLA) ID(IZUUSER) ACCESS(READ)
```

z/OSMF automatically manages the authorization of non-authenticated guests (not logged in) and authenticated guests (logged in, but are not defined to a z/OSMF SAF security group). By default, a non-authenticated guest user can access the z/OSMF Welcome task and access the default links. An authenticated guest can access everything a non-authenticated guest can, and also view the online help.

To authorize authenticated guest users to your task, in the plug-in property file, set the *task-authenticated-guest-eligible* property to *true*. Extending task authorization to users who are not logged into z/OSMF is not supported.

Actions for security update

Changes to your security setup require applicable refreshes of the security product and a restart of the z/OSMF server for them to take effect.

Chapter 4. Preparing software for cloud provisioning

This topic describes how to prepare files that are required to make software available to exploit IBM Cloud Provisioning and Management for z/OS.

IBM Cloud Provisioning and Management for z/OS allows software consumers, from a selection of software services, to quickly provision and deprovision software as needed. For more information, see "Cloud provisioning services" on page 46 and the online help for the Cloud Provisioning tasks in z/OSMF.

Software is provisioned from a software services template. A software services template requires the following files.

Workflow definition file

A workflow definition file is the primary XML file that defines the workflow that performs the provisioning. The workflow definition file includes information about the workflow, such as name and version, as well as step and variable definitions. For information about workflow definition files, see Chapter 2, "Creating workflow definitions for z/OS," on page 591.

Variable input file

A workflow variable input file is a properties file that is used to specify values for one or more of the input variables that are defined in the workflow definition. For information, see Chapter 2, "Creating workflow definitions for z/OS," on page 591.

Action definition file

An action definition file describes the actions that can be performed against the provisioned software, which is known as a software services instance. For more information, see "Actions definition file" on page 745.

Documentation files

A documentation file is an optional text or PDF file that provides information that is important to provisioning the software. For example, it might describe the workflow and other files, and describe requirements for using them to provision software. There can be one documentation file for administrators, who create the software services template and prepare the software for provisioning, and one for consumers, who use the published software services template to provision the software. The document for administrators might indicate, for example, whether a network resource pool or WLM resource pool is required.

Manifest, or template source file

A manifest file is optional. It provides a shortcut when a user creates the software services template using the z/OSMF Software Services task. Rather than specifying each of the files (workflow definition, input variable file, action, and documentation) individually, the user can specify just the manifest file, then click **Load** to supply values for the other files.

The file must be in Java™ property file format:

- Each entry is a single line, in *property=value* or *property:value* format.
- The \ character is a continuation character, so that a value can span lines.

- For newline, carriage return, and tab, use \n, \r, and \t.
- Comment characters are ! and #. Lines that start with those characters are ignored.

The fields in the manifest file are:

workflow-definition-file
Name of the workflow definition file

workflow-variable-input-file
Name of the workflow variable input file

action-definition-file
Name of the action definition file

description
Brief description of the workflow. This is optional.

admin-documentation-file
Name of the file that describes the workflow and other files. This file is intended for an administrator who will prepare a software services template that consumers can use to provision the software. This is optional.

admin-documentation-type
File type of the administrator documentation file: txt or pdf. This is optional, and valid only if admin-documentation-file is specified.

cconsumer-documentation-file
Name of the file that describes the workflow and other files, intended for a consumer who will use the software services template to provision the software. This is optional.

cconsumer-documentation-type
File type of the consumer documentation file: txt or pdf. This is optional, and valid only if consumer-documentation-file is specified.

The following is an example of a manifest file.

```
# provision.mf
#
# Manifest file to be used when adding a template for provisioning an MQ for z/OS Queue Manager.
#
#  <copyright
#      notice="lm-source"
#      pids="@PID###@"
#      years="2015,2016"
#      crc="3073404564">
#      Licensed Materials - Property of IBM
#
#      @PID###@
#
#      (C) Copyright IBM Corp. 2015, 2016 All Rights Reserved.
#  </copyright>
#
# Provision Queue Manager workflow file (steps to provision a Queue Manager)
workflow-definition-file:provision.xml
# Provision Queue Manager workflow variables properties file (properties to be used when provisioning a Queue Manager)
workflow-variable-input-file:workflow_variables.properties
# Queue Manager actions file (defines the actions that can be performed against a Queue Manager)
action-definition-file:qmgrActions.xml
# Provision Queue Manager workflow description
description:This workflow provisions an MQ for z/OS Queue Manager
# Provision Queue Manager readme file
admin-documentation-file:mqaas_readme.pdf
# Provision Queue Manager readme file type
admin-documentation-type:pdf
```

Actions definition file

An actions definition file has XML syntax and conforms to the rules of the actions schema. The schema defines the required and optional properties (XML elements and attributes). It imposes constraints on the order in which the elements are specified, and on the values that can be specified for each element and attribute.

The schema file is UTF-8 encoded.

If you are developing an actions definition file, you require access to the schema, and therefore access to the z/OS system on which z/OSMF is installed.

The primary XML file must start with a processing instruction (in column 1 of line 1) for the XML processor. This instruction defines the version of XML used and the encoding of the file. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The remaining elements are as follows.

<actionList>
Is the root element. It contains the actions definitions.

<action>
Contains an action definition. There must be 1 - 50 actions in an actions definition file. The action contains either a command, workflow, or instruction element. The attributes are:

- name, which specifies the name of the action
- deprovision, which accepts true or false, to indicate whether the action is for deprovisioning. There must be at least one deprovision action.

<command>
Contains a command definition. It contains the following elements.

<commandValue>
Command to be issued. This is required.

<runAsUser>
User ID under which the action is to be performed. This is optional. The attribute is substitution, which accepts true or false.

<approver>
A user ID, or a list of user IDs separated by blanks. At least one user ID must approve the action before it is performed on behalf of the user ID that is specified with the runAsUser element. To specify multiple required approvers, use multiple approver elements (up to 12). The approver element is optional. If it is specified, then the runAsUser element is required.

<unsolkey>
Key to search for in the unsolicited messages, for a command-type action.

<solkey>
Key to search for in the solicited messages command response, for a command-type action.

<detectTime>
Time in seconds to search for the unsolkey in the unsolicited messages. Also, the minimum time before a command response is checked for after the command is submitted for execution.

<workflow>
Contains a workflow definition, consisting of the elements that follow.

<wfDefFile>
Workflow definition file path. This element is required. The maximum length of the file path is 1024 characters.

<wfVarInFile>
Workflow variable input file path. This element is optional. The maximum length of the file path is 1024 characters.

<wfVar>
Value of a workflow variable. This element is optional. Up to 1500 variables are allowed. It includes a name attribute, which defines the name of the variable. The name must be unique.

<instruction>
Defines an instruction.

Example

The following example shows an action definition.

```
<?xml version="1.0" encoding="utf-8"?>
<actionList>
        <action name="workflow1">
                <workflow>
                       <wfDefFile>workflow1.xml</wfDefFile>
                </workflow>
        </action>
        <action name="workflow2">
                <workflow>
                       <wfDefFile>workflow2.xml</wfDefFile>
                </workflow>
        </action>
        <action name="instructions1">
                <instructions>The instructions</instructions>
        </action>
        <action name="command1">
                <command>
                   <commandValue>d iplinfo</commandValue>
                </command>
        </action>
        <action name="deprovision" deprovision="true">
                <workflow>
                       <wfDefFile>deprovision.xml</wfDefFile>
                </workflow>
        </action>
</actionList>
```

Appendix A. Enabling tracing for the z/OS jobs REST interface

For diagnostic purposes, your installation might be asked by IBM Support to enable tracing for the z/OS jobs REST interface. This topic provides instructions for enabling several commonly-used traces.

Your installation can trace the use of the z/OS jobs REST interface on the z/OSMF system. Also, you can trace a variety of JES related activities, which can result from the use of the z/OS jobs REST interface services.

Understand that tracing carries a performance cost. Do not activate tracing for z/OSMF unless directed to do so by IBM Support.

Tracing the z/OS jobs REST interface services

To trace the use of z/OS jobs REST interface services on the z/OSMF system, use the **MODIFY** command with the `LOGGING` option. Your user ID must be permitted to enter this operator command.

The command has the following format:

```
f server-name,logging='com.ibm.zoszmf.restjobs.*=all'
```

where:

server-name
: Is the server for your z/OSMF configuration. Set this value to the job name of the z/OSMF server, which is `IZUSVR1`, by default.

com.ibm.zoszmf.restjobs.*=all
: Is the trace specification for the z/OS jobs REST interface.

Enter the command from the operator console. The command output is displayed in the operator console and in the z/OS system log.

Your changes take effect immediately and remain in effect while the server is running. Your changes are discarded when the server is restarted, and the previous settings are used.

To end this level of tracing and revert to the previous setting, enter the command again, and specify "reset" as the trace specification, for example:

```
f izusvr1,logging='reset'
```

Tracing the JES related activities for your programs

For callers of the z/OS jobs REST interface, your installation can trace the JES related activities that can occur on behalf of program requests.

Specifically, you can trace the following JES related activities:

- Usage of the following subsystem interface (SSI) function codes:
  - Extended status function call (SSI function code 80), which allows a user-supplied program to obtain detailed status information about jobs and SYSOUT in the JES queue
  - Modify job function call (SSI function code 85), which allows a user-supplied program to modify job properties and to manage memory associated with the request.
- VSAM related activities.
- JES symbolic parameter substitutions.

- HSM recall activities.

To capture this type of information, you must add the appropriate trace specifications to the z/OSMF bootstrap.template file, which ensures that the server is started with the proper traces enabled. z/OSMF writes the trace output to files in the z/OSMF logs directory.

To start this type of tracing, do the following:

1. Locate the z/OSMF `bootstrap.template` file. By default, the location is: `/etc/zosmf/servers/zosmfServers/bootstrap.template`
2. Save a copy of the existing bootstrap.template file as a back-up.
3. Edit the `bootstrap.template` file, as needed:
   - Add the property `zosjes.logging=t` to capture information about the following activities:
     - Usage of the extended status function call (SSI function code 80)
     - VSAM related activities.
   - Add the property `izurestjobs.logging=t` to capture information about the following activities:
     - Usage of the modify job function call (SSI function code 85)
     - JES symbolic parameter substitutions
     - HSM recall activities.

   A portion of the file is shown in Figure 369.

```
# Licensed Materials - Property of IBM
#
# "Restricted Materials of IBM"
#
# Copyright IBM Corp. 2013 All Rights Reserved.
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with
# IBM Corp.
#
# ------------------------------------------------------------
#

izu.hostname=*
izu.https.port=443

# Trace options follow...
zosjes.logging=t
izurestjobs.logging=t
```

Figure 369. Bootstrap properties for z/OSMF

4. Save the `bootstrap.template` file.
5. Restart the z/OSMF server and resume z/OSMF operations.

Your changes will take effect immediately and are maintained across z/OSMF server restarts.

To work with the z/OSMF log files, you require a user ID with z/OSMF administrator authority (that is, a user ID defined to the z/OSMF administrator security group).

For information about how to enable other trace options for z/OSMF, and how to work with z/OSMF log files, see *IBM z/OS Management Facility Configuration Guide*.

Appendix B. Creating product information files for the Software Management task

A *product information file* is a flat file, such as a text file, that contains information about one or more products. This information includes, for example, the product announce date, general availability date, and end of service date. You can create your own product information files or obtain them from a provider, such as IBM, another vendor, or a third party.

z/OSMF displays data from product information files in several views in the Software Management task. For example, this information is displayed in the Products page, the Products, Features, and FMIDs page, and the End of Service report.

Syntax for product information files

To be processed by z/OSMF, product information files must be formatted as JSON data and have the following syntax:

```
{
  "Version": "date-modified",
  "Products":
  [
    {
      "prodName": "product-name",
      "prodId": "product-identifier",
      "prodVRM": "version-release-modification",
      "GAAnnounceDate": "date-announced",
      "GADate": "general-availability-date",
      "URL": "URL",
      "EOSDate": "end-of-service-date",
      "country": "country"
    }
  ]
}
```

where,

date-modified
: Date the file was created or last updated. The date must have the format YYYY-MM-DD. The date is required.

product-name
: Name of the product. The name is optional, and is not used by z/OSMF. To omit the product name, exclude the field, type `null` as the value, or set the value equal to an empty string.

product-identifier
: Identifier of the product. The product ID is required.

version-release-modification
: Version, release, and modification level of the product. The value has the format *VV.RR.MM*, where *VV* is the two-digit version, *RR* is the two-digit release, and *MM* is the two-digit modification level. The version, release, and modification level are required.

date-announced
: Date the vendor publicly announced the details of the product. The date must have the format YYYY-MM-DD. The date is optional. To omit the date, exclude the field or type `null` as the value.

general-availability-date
Date that a version or release of the product is available to all users. The date must have the format YYYY-MM-DD. The date is optional. To omit the date, exclude the field or type `null` as the value.

URL URL that links to additional information about the product. This information can include, for example, product life cycle dates, product highlights, planning information, and technical descriptions. The URL is optional. To omit the URL, exclude the field, type `null` as the value, or set the value equal to an empty string.

end-of-service-date
Last date on which the vendor will deliver standard support services for a given version or release of the product. This date is the general end of service date. It does not account for lifecycle extensions. The date must have the format YYYY-MM-DD. The date is optional. To omit the date, exclude the field or type `null` as the value.

country
Country for which the end of service date is applicable. The country is optional. To omit the country, exclude the field, type `null` as the value, or set the value equal to an empty string.

The information for each product must be contained within separate braces ({ }) inside the brackets ([ ]), and each set of braces must be comma separated. For a sample file that contains the information for two products, see Figure 370 on page 751.

Sample product information file

```
{
  "Version": "2011-06-30",
  "Products":
  [
    {
      "prodName": "z/OS",
      "prodId": "5694-A01",
      "prodVRM": "01.10.00",
      "GAAnnounceDate": "2008-08-05",
      "GADate": "2008-09-26",
      "URL": "http://www-03.ibm.com/systems/z/os/zos/",
      "EOSDate": "2011-09-30",
      "country": "US"
    },
    {
      "prodName": "z/OS",
      "prodId": "5694-A01",
      "prodVRM": "01.13.00",
      "GAAnnounceDate": "2011-07-12",
      "GADate": null,
      "URL": "",
      "country": "US"
    }
  ]
}
```

Figure 370. Sample product information file for the Software Management task

Working with the IBM product information file

The product information file that IBM supplies for System z® software is located at the following URL: http://public.dhe.ibm.com/services/zosmf/JSONs/IBMProductEOS.txt .

To load the contents of the file into z/OSMF, do one of the following:

- Load directly from the URL.
- Manually download the file at the URL to your local workstation.
- Manually download the file at the URL to a z/OS data set or UNIX file that the primary z/OSMF host system can access.

When transferring the file from a workstation to a z/OS data set or UNIX file, transfer the file in binary format. To avoid errors, do not convert the file to the EBCDIC character set.

After you store the file in your desired location, to retrieve its contents, complete the steps provided in the *Retrieving product information from product information files* topic in the z/OSMF online help.

Appendix C. Portable Software Instance

A software instance is a collection of data sets containing installed software, and other data sets that may be associated with that installed software. The software may be SMP/E managed, in which case the collection of data sets also contains, and is described by, one or more SMP/E target and distribution zone pairs, defined by a single global zone.

A portable software instance is, exactly as the name implies, a portable form of a software instance,which can be used to simplify distribution of a software instance across a network, and can be deployed by the z/OSMF Software Management task. A portable software instance is a set of portable archive files created by the SMP/E GIMZIP service routine for each of the data sets defined to the software instance, including SMPCSI data sets with all associated SMP/E managed target and distribution libraries, and a descriptor file to describe in detail the entire originating software instance.

Portable Software Instance Descriptor File

The portable software instance descriptor file contains detailed information to describe the content of the originating software instance. It contains the information required by the z/OSMF Software Management task to perform a deployment operation on the content of the portable software instance. The portable software instance descriptor file has a file name of IZUD00DF.json, and is created by z/OSMF during the operation of an Export action on a software instance.

The content of the portable software instance descriptor file is shown in Portable software instance descriptor file.

```
{"izud.pswi.descriptor":
 {
  "version":"pswi-descriptor-version"
  "created":"yyyy-mm-ddThh:mm:ssZ",
  "gimpaflocation":"relative-path-for-GIMPAF.XML-file",
  "name":"software-instance-name",
  "description":"software-instance-description",
  "globalzone":"csi-data-set-name",
  "zones":[{
    "name":"zone-name",
    "type":"zone-type",
    "related":"related-zone-name",
    "csi":"csi-data-set-name"
    }]
  "datasets":[{
    "dsname":"data-set-name",
    "volumes":["volume-serial"],
    "storclas":"storage-class",
    "dstype":"data-set-type",
    "tracks":"allocated-tracks",
    "zonedddefs":[{
      "zone":"zone-name",
      "dddefs":[{"dddef":"dddef-name", "path":"unix-directory"}]
      }],
    "mountpoint":"UNIX-path",
    "isextendedformat":true | false",
    "recfm":"record-format",
    "lrecl":"logical-record-length",
    "blksize":"block-size",
    "used":"used-tracks-percent"
    "extents":"allocated-extents",
    "dscategory":["data-set-category"],
    "archid":"gimzip-archive-id",
    }]
  "smpeproducts":[{
    "prodname":"product-name",
    "prodid":"product-id",
    "release":"vv.rr.mm",
    "vendor":"vendor-name",
    "url":"product-url",
     "srels":["srel"],
     "prodsups":[{
     "prodid":"product-id",
     "release":"vv.rr.mm"
      }],
    "features":[{
      "featname":"feature-name",
      "featid":"feature-id",
      "fmids":["fmid-name"]
      }]
    }
 }
}
```

Figure 371. Portable software instance descriptor file

Where:

- **version** the version of the portable software instance descriptor. The initial version is 1.
- **created** the date and time when the portable software instance was created, in ISO 8601 format. For example, yyyy-mm-ddThh:mm:ssZ.
- **gimpaflocation** the relative path to the GIMPAF.XML file for this portable software instance, relative to the location of the portable software instance descriptor file.
- **name** the name for the originating software instance.

- **description** the description of the originating software instance. This is an optional property.
- **globalzone** the name of the CSI data set that contains the global zone. This is an optional property, specified only if the originating software instance describes SMP/E managed software.
- **zones** the list of SMP/E zones from the originating software instance. This is an optional property, specified only if the originating software instance describes SMP/E managed software.
- **name** the zone name.
- **type** the type for the zone, global, target or dlib.
- **related** the name of the zone's related zone, if any.
- **csi** the name for the CSI data set that contains the zone.
- **datasets** the list of data sets from the originating software instance.
- **dsname** the originating data set name.
- **volumes** the list of volume serials where the originating data set resided.
- **storclas** the name of the storage class where the originating data set resided. This is an optional property.
- **dstype** the type for the data set. Can be one of the following types:
  - HFS – Hierarchical file system.
  - PDS – Partitioned data set.
  - PDSE – Partitioned data set extended.
  - SEQ – Sequential data set.
  - VSAM – VSAM data set.
  - ZFS – zSeries file system.
- **tracks** the number of 3390-device equivalent tracks (56664 bytes/track)allocated to the data set.
- **zonedddefs** the list of SMP/E zones and DDDEF entries that reference the data set. This is an optional property, specified only if the originating software instance describes SMP/E managed software, and if the subject dataset is referenced in the SMP/E zones.
- **zone** the name of an SMP/E zone that contains one or more DDDEF entries for the data set.
- **dddefs** the list of DDDEF entries that identify the data set.
- **dddef** the name of the DDDEF entry.
- **path** the UNIX directory identified in the DDDEF entry. Null if the DDDEF entry identifies a data set.
- **mountpoint** the mount point for the originating UNIX file system data set. Null if the DDDEF entry identifies a data set instead of a UNIX directory. This is an optional property, specified only if the originating software instance describes SMP/E managed software, and if the subject dataset is referenced in the SMP/E zones by a DDDEF entry with a UNIX directory.
- **isextendedformat** indicates, true or false, if the data set is an extended format sequential data set.
- **recfm** the record format. The record format can be any valid combination of the following codes:
  - A – ASA printer control characters.
  - B – Blocked records.
  - F – Fixed-length records.
  - M – Machine code printer control characters.
  - S – Standard (for F) or spanned (for V); used only with sequential data sets.
  - T – Track-overflow feature.
  - U – Undefined format records.
  - V – Variable-length records.
- **lrecl** the logical record length.
- **blksize** the block size, in bytes.

- **used** the percentage of allocated tracks used, expressed in whole numbers, not rounded. If any track is used, the minimum percentage is 1. If the data set is a PDSE, the percentage refers to the percentage of allocated pages used.
- **extents** the number of extents allocated to the data set.
- **dscategory** List of categories for how the data set is used. Can be one or more of the following:
  - target – SMP/E managed target library, or SMP/E control dataset associated with a target zone.
  - dlib – SMP/E managed distribution library, or SMP/E control data set associated with a dlib zone .
  - global – SMP/E control data set associated with the global zone.
  - smp – SMP/E control data set.
  - other – None of the above.
- **archid** the archive ID value, produced by the GIMZIP service routine and specified in the GIMPAF.XML file, to identify the portable archive file for the data set.
- **smpeproducts** the list of software products installed in the originating software instance that are managed by SMP/E. This is an optional property.
- **prodname** the name for the product.
- **prodid** the identifier for the product.
- **release** the version, release, and modification level for the product, in this format: vv.rr.mm.
- **vendor** the name for the product's vendor. This is an optional property.
- **url** the URL that links to additional information about the product. This is an optional property.
- **srels** the system or subsystem releases on which the subject product can be installed.
- **prodsups** the list of products that are superseded by the subject product. This is an optional property.
- **features** the list of features for the subject product.
- **featname** the name for the feature.
- **featid** the identifier for the feature.
- **fmids** the list of FMIDs for the subject feature. It is also the list of FMIDs in the originating software instance. This list includes all FMIDs associated with one or more products and features, and all FMIDs associated with no products or features.

Appendix D. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that

have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used

zero or more times; that is, it is optional and can be repeated. For example, if you hear the line `5.1* data area`, you know that you can include one data area, more than one data area, or no data area. If you hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loopback line in a railroad syntax diagram.

`+ indicates a syntax element that must be included`

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST, and 2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the `*` symbol, the `+` symbol can repeat a particular item if it is the only item with that dotted decimal number. The `+` symbol, like the `*` symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

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

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

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

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

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

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

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

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

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

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

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

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

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

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

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

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

COPYRIGHT LICENSE:

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

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

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 at Copyright and Trademark information (http://www.ibm.com/legal/copytrade.shtml).

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Intel is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names might be trademarks or service marks of others.

Index

G

H

I

J

K

L

M

N

O

P

Product Number: 5610-A01

Printed in USA

SA32-1066-08