HP Operations Orchestration

For Windows ® and Linux

Software Version: 9.07

Software Development Kit Guide

Document Release Date: January 2013

Software Release Date: January 2013

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Welcome to the Operations Orchestration SDK

The **Hewlett-Packard Operations Orchestration Software Development Kit** (OO SDK) contains documentation, tools, libraries, and code samples for developers and IT professionals who want to:

- Learn best practices for designing flows, steps, and operations.
- Create **IActions** to run OO operations through a **Remote Action Service** (RAS).
- Find and run flows from outside Central.
- Run repository functions from outside Studio.
- Package new and updated content for distribution on Central and RAS servers.
- Inspect a repository.
- Automate flow testing.
- Debug OO client/server problems.

In OO: How to find Help, PDFs, and tutorials

The OO software documentation set is contains:

- Help for Central

  Central Help provides information on:

  - Finding and running flows.
  - Configuring OO functioning (for OO administrators).
  - Generating and viewing the information available from the outcomes of flow runs

  The Central Help system is also available as a PDF document in the OO home directory, in the **Central\docs** folder.

- Help for Studio

  Studio Help instructs flow authors at varying levels of programming ability.

  The Studio Help system is also available as a PDF document in the OO home directory, in the **Studio\docs** folder.

- Animated tutorials for Central and Studio

  OO tutorials can each be completed in less than half an hour and provide basic instruction on:

  - finding, running, and viewing information from flows (in Central).
  - modifying flows (in Studio).

The tutorials are available in the OO home directory, in the **Central\tutorials** and **Studio\tutorials** folders.

- Self-documentation for OO operations, flows, and Accelerator Packs.

  Self-documentation is available in the descriptions of the operations and steps that are included in the flows.

Download SDK

To locate and download the SDK Installation Package:

1. Go to **https://hpln.hp.com/group/operations-orchestration**
2. Click **Other Files**
3. Expand **HP Operations Orchestration 9.00**
4. Expand **HP Operations Orchestration 9.07 SDK**
5. Open the **zip** file.

You can place the SDK in any location on a Central or RAS server. The code samples in this guide can be placed anywhere and you can use most development tools to point to the code, import and use it as if it was on Central.

SDK Contents

In this guide, the folder in which you install the SDK is referred to as the **OO SDK home directory**. The basic folder structure of the OO SDK home directory is:

- docs\ folder
  - javadocs\ folder
  - SDKGuide.pdf
- lib\ folder
  - ContentCommons-9.00.jar
  - dharma-commons-9.00.jar
  - JRAS-sdk-9.00.jar
  - wscentral.dll
  - WSCentralService.jar
- samples\ folder
- AutoTest.jar
- ContentPackager.jar
- JRSFlowInvoke.jar
- RepoInspector.jar
- RepoTool.jar

- RSFlowInvoke.exe
- sdk_contents.txt

The SDK contains the following components:

- SDKGuide.pdf

  The documentation for the entire SDK. It includes conceptual information, descriptions and step-by-step instructions for using tools, command syntaxes, class and method syntaxes, code examples, and code samples. The **SDKGuide.pdf** file is located in the OO SDK home directory, in the **docs** folder.

- IAction interface, methods, and classes

  IAction interface, methods, and classes that allow you to author Java and .NET IActions - code that implements OO operations through a **Remote Action Service** (RAS). The IAction interface, methods, and classes are located in the OO SDK home directory, in the **lib** folder.

- **Application Programming Interface** (API) documents

  Javadocs for both the JRAS and Central. The javadocs are located in the OO SDK home directory, in the **docs\javadocs** folder.

- **WSCentralService Simple Object Access Protocol** (SOAP) API

  The **WSCentralService** API Java and .NET classes and interfaces are located in the OO SDK home directory, in the **lib** folder. The certificates, **keystore**, **WSDL**, and code samples are located in the OO SDK home directory, in the **samples** folder.

- Samples

  IAction Java sample code and WS Central Service SOAP API sample code.

- AutoTest.jar

  A utility that allows you to run automated tests. **AutoTest.jar** is located in the OO SDK home directory.

- ContentPackager.jar

  Tools and commands that allow you to package and install OO content updates. **ContentPackager.jar** is located in the OO SDK home directory.

- RepoInspector.jar

  A utility that allows you to check the repository. **RepoInspector.jar** is located in the OO SDK home directory.

- RepoTool.jar

  A utility that allows you to perform a number of repository functions from outside Studio. The **RepoTool.jar** utility is located in the OO SDK home directory.

- RSFlowInvoke.exe and JRSFlowInvoke.jar

  The Windows and Java Versions of a utility that allows you to find and run OO flows outside Central. You can do this from a command line, an application that uses a command line, a script, or a batch file. **RSFlowInvoke.exe** and **JRSFlowInvoke.jar** are located in the OO SDK home directory.

About the SDK Guide

The SDK Guide provides information on:

- The folder structure of the installed SDK and the SDK contents.
- How OO content is organized in Studio, provides guidelines for flow layout and naming conventions, and best practices for creating flows, steps, and operations.
- How to use the IAction interface, methods, and classes to create Java and .NET IActions - OO operations that are implemented through a RAS. It also explains how to load your IActions into Studio and debug them.
- How you can manage flows outside Central, using:
  - URLs created in Central.
  - Command-line tools that access the **REST** service: **Wget.exe**, **RSFlowInvoke.exe**, and **JRSFlowInvoke.jar**.
  - The **WSCentralService SOAP API**.
- Working with repositories from outside Studio: how to use the **RepoTool.jar** utility to perform repository functions from outside Studio.
- Packaging content: how to use the **ContentPackager.jar** utility to package updated content and publish it to Central and RAS servers in your network.
- How to use the **RepoInspector.jar** utility to check a repository.
- Automating flow testing: how to use the **AutoTest.jar** utility to stress test your flows.
- Debugging OO client/servers problems: how to allow HTTP connections to Central and RAS for debugging purposes.

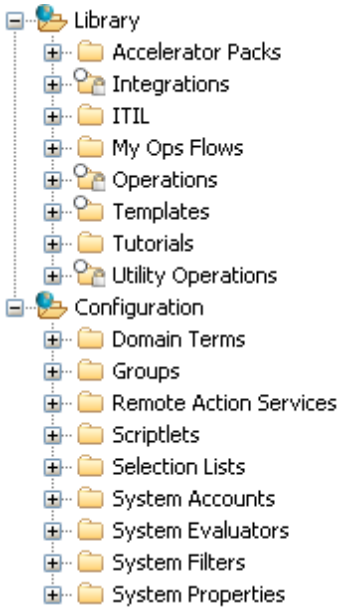
Chapter 1

Style guidelines and best practices

Following style guidelines and best practices for creating content (operations and flows) in OO enables content and quality assurance engineers, field engineers, and customers to create flows quicker and more efficienty. Follow these guidelines for any content you create and submit to the OO content community.

How default OO content is organized in Studio

Default OO content consists of all the flows and operations that come with your installation of OO, contained in folders in the Studio **Library**.



Folders in Studio Library

The following describes the Studio Library folders that contain default content:

Folder	Folder Contents
Accelerator Packs	Flows, organized into folders by technologies, designed to solve common IT problems. For most networks, these flows: • Perform complex health checks, triage, diagnosis, or remediation. • Gather one or more pieces of data and display it to the user, or simply acknowledge alerts, gather data, and place it into a ticket. The flows at the top level of an Accelerator Pack are usually full health check, triage, diagnosis, and remediation flows.

Folder	Folder Contents
Integrations	Operations and flows you can use to integrate OO with other enterprise management software products, such as Hewlett-Packard Network Node Manager and BMC Remedy. Because the enterprise software products used in your data center can be highly customized, you might need to create custom flows to use these operations.
ITIL	Flows that automate integrations with other enterprise-level software in accordance with the Information Technology Infrastructure Library (ITIL) specifications, such as **Change Management**.
Operations	General-purpose operations and flows that work with common technologies. These operations are sealed and cannot be changed once you have installed OO Central. The flows in the **Operations** folder and its subfolders are meant to be used as subflows. The flows that are meant to be run on their own are in the **Accelerator Packs** folder.
Templates	Templates that provide steps for flows that perform certain frequently used tasks. For example, the **Restart Service** template restarts a service, so you could use it in a flow that includes this task.
Utility Operations	Operations and subflows that gather and display data, replace simple command-line operations, manipulate and analyze data, provide structure to flows, and perform other tasks that are not specific to a certain technology.

Note: When you install OO, the **My Ops Flows** folder is empty. When you create flows from templates, OO automatically stores them in the **My Ops Flows** folder. You can also store in this folder flows that you create.

Best practices for flows

The following best practices will make it easier for customers to use the flows you create:

- Best practices for flow inputs:

  Ideally, input values used by flow steps are supplied by flow inputs and passed to the steps by flow variables. This may not always be practical. For instance, a user might need to enter an input in response to a prompt somewhere in the flow run.

  In general, flow authors should assume that a user will begin a flow and then start another task while the flow is running. Assigning as much data as possible to flow inputs also simplifies making changes to the flow.

- Best practices for flow descriptions:

  To help Central users who use your flows, and authors who use them as subflows to create other flows, add information to the flow's **Description** tab as explained in the following paragraphs. If you create multiple flows or operations that interact with the same technology,

group them into a single folder and provide this information in the folder's **Description** tab. This is the practice for default OO content.

Putting this information on the **Description** tab makes it available to authors and Central users through the **Generate Documentation** feature. For more information on the **Generate Documentation** feature, see the *Studio Authoring Guide* or the *Studio online Help* system.

Add to the **Description** tab a description of what the flow does and the information needed to successfully use the flow and obtain useful information from the flow results.

- Any special requirements or changes that are necessary for the flow to run automatically (on a schedule or started from outside Central).
- Limitations to the flow's usage, such as:

```
Limitations:

This flow only works:

-- On Windows 2003 or later.

-- If the Windows Telnet Service is enabled.

-- If RAS is installed on a host running Microsoft Operations Manager.
```

- Which flow inputs are required, and information on where authors can find the required input data and on the data format.
- The result fields, including a description of the data supplied in each result field.
- The flow responses, including the meaning of each response.
- Any additional implementation notes, such as:
  - Supported platforms or applications, including version information.
  - Application or Web service APIs that the flow interacts with. This is particularly important for flows that require a RAS to run, because the RAS operation can hide this information from the author or the user of the flow.
  - Other environmental or usage requirements.

- Other best practices for flows:
  - A flow performing triage, diagnosis, or remediation should first verify if a problem exists.
  - A flow sending a notification to the user should use notification subflows, which enable the flow author to choose from several means of notifying the user.

    For instance, the **Web site Health Check** flow uses the **Notify** subflow. Once the user configures the **Web site Health Check** flow to his or her e-mail and ticketing systems, all flows using **Web site Health Check** will send notifications correctly.
  - Supply a description for all transitions (annotate) in a top-level parent flow.

    These transition descriptions should describe what happened in the step that preceded the transition. In Central, the **Results Summary** for the run displays the description for each transition, and so provides a running, high-level account of what took place in the flow run. You don't need to annotate transitions in a subflow unless it is critical to see the data during a run in Central.

Best practices for steps

To streamline the steps in a flow, use the following best practices:

- Steps do not generally require descriptions, because the transition description of the step's response tells what happened in the step.
- An operation can provide many results. At step level, assign to flow variables only the results needed by the flow.
- If a step or transition needs the exact error that came back from an operation, create a step result that captures the error code, and assign the error code to a flow variable.
- To assign information to a flow variable, use the step's **Results** tab. Filters on the results greatly enhance your flexibility in obtaining data from step results.
- Any time a step makes a modification to the IT environment, consider recording the data for **Dashboard** reporting in Central. If a change is made, reporting information should be recorded on the next step following the success transition. This often means that reporting information is recorded on flow return steps.

Best practices for operations

To help authors create flows using the operations you create, use the following best practices:

- Add information to the flow's **Description** tab as explained in the following paragraphs. If you create multiple flows or operations that interact with the same technology, group them into a single folder and provide this information in the folder's **Description** tab. This is the practice for default OO content. Note that putting this information on the **Description** tab makes it available to authors and Central users through the **Generate Documentation** feature. For more information on **Generate Documentation**, see the *Studio Authoring Guide* or the *Studio online Help* system.

  Add to the **Description** tab a description of what the operation does and the information needed to successfully use the operation and obtain useful information from the operation results:

  - Which operation inputs are required, and information on where authors can find the required input data and on the data format.
  - The result fields, including a description of the data supplied in each result field.
  - The operation responses, including the meaning of each response.
  - Any additional implementation notes, such as:
    - Supported platforms or applications, including version information.
    - Application or Web service APIs that the operation interacts with. This is particularly important for operations that require a RAS to run, because the RAS operation can hide this information from the author or the user of the operation.
    - Other environmental or usage requirements.
  - Use the following template as the basis for operation descriptions:

```
A brief operation description.

Inputs:
    Input1 - Information about the first input.
    Input2 - Information about the second input.
    Input3 - Information about the third input.

Results:
    Result0 - Information about the primary result.
    Result1 - Information about the first additional result.
    Result2 - Information about the second additional result.

Responses:
    Response1 - Information about the first response.
    Response2 - Information about the second response.
```

- Do not make copies of the sealed operations, such as the ones in the **Operations** folder. Instead, make changes to the steps that you created from sealed operations.
- By default, operations should use and set flow variables for inputs that are used repeatedly in a particular flow. For example, multiple operations in a flow might need the `host`, `username`, and `password` inputs to get information from a server or the port of a mail server. Assigning those values to flow variables used in various steps requiring such data simplifies flow maintenance and makes it easier to adapt to various situations.

  In contrast, the subject line of an e-mail is probably different for each step that requires an e-mail subject line. Therefore, the subject line is probably not a good candidate for being provided from a flow variable.
- Avoid creating multiple operations that run the same command. For example, you can get both packet loss and maximum latency from a ping operation. Rather than creating multiple operations that use the `ping` command, a better practice is to capture both pieces of information in one step, using multiple outputs of just one ping operation.

  Exceptions to this principle are operations that are extremely generic. For example, an operation that runs a Windows Management Instrumentation (WMI) command. It is better to create WMI command operations that are specific to particular functions, instead of a single operation that has a very generic input for the WMI command and very generic outputs.
- For capturing data from the output stream of a command, using result filters is better than using a scriptlet. There are several reasons:
  - Result filters are accessible and immediately visible on the **Results** tab editor rather than residing separately, as scriptlets do, on the **Scriptlets** tab.
  - Scriptlets are more difficult for non-programmers to maintain.
  - If an operation result is removed, the result filters are automatically invalidated. In case of scriptlets, after deleting the result that the scriptlet manipulates, any scriptlets that the author fails to remove remain and might cause errors in the flow.
  - If you need a scriptlet for result data processing, you can use a scriptlet filter.
- Most operations should have two responses: `success` and `failure`. Using a small number of responses eases flow creation and understanding. Multiple responses, based on different failure types, should only be used when there are obvious distinct paths to follow or there are circumstances where an outcome may only be a failure because of the situation (such as a redirection response to an **HTTP Get**).

However, don't force this principle when it doesn't make sense. For example, an operation that gets data and checks a threshold may require three responses, none of which being a `success` response: `failure`, `over threshold`, and `under threshold`.

- The default response for an operation should be `failure`. This way, an incomplete operation is shown as a failure during flow debugging and points the author to the problem before the flow goes into production.

Naming convention guidelines

Using the following naming conventions significantly helps authors debug or modify flows and operations:

- Use Title Case (first letter capitalized for all except helper words like 'a', 'the', 'and', 'by', 'for') for:
  - Items in the **Library** (flows, folders, and operations) and items in the **Configuration** folder.

    For example: **Reboot a Server**, **Check the Log Files**, and, in the **Configuration\Domain Terms** folder, **CI Minor Type**.
  - Step names.
- Use lower case for operation responses (spaces are permissible).

  Example: `failure`, `success`, `over threshold`
- Use camel case (first letter of the name is lower-case; subsequent first letters of words contained in the name are upper-case) for:
  - Input names, for example `protocol` and `messageNumber`
  - Output names, for example `hopCountThreshold`
  - Result names, for example `aclData`
  - Flow variable names, for example `aclData` and `userId`

  No space or other non-alphanumeric characters are allowed in camel case names.
- Common input names occur across many operations and steps. To ease authoring using operations that are available immediately upon installing OO, the following input names are used in OO content:
  - `host` - For Windows, the host is the machine on which the operation works. For example, the host from which you are getting a performance counter or on which you are restarting a service. For secure shell (SSH) operations, the host is the machine on which the command is running.
  - `username` - The name of the account to use for logging on to the machine.
  - `password` - The password to use to log on to the machine.

  Use the following template to list these inputs in an operation description:

```
Inputs:
    host - The host to run the command against.
    username - The user name used to log on to this machine.
    password - The password associated with the <username> input value.
```

- Other common input names include:
  - `mailHost` - The host machine from which an e-mail is sent.
  - `target` - When the host affects another system, the affected system should be called the target. For example, if you SSH to server1 to run a ping against server2, then the host is server1 and the target is server2.

Chapter 2

Authoring IActions

This chapter defines IActions, Remote Action Service (RAS), and explains how to:

- Use the IAction interface and methods to author Java and .NET IActions.
- Load your IActions into OO Studio.
- Debug your IActions.

It also provides code examples, and useful Java and .NET Commons Library classes that can help you develop IActions.

What is an IAction?

An IAction is the code implementing an operation through a RAS operation. The RAS is a service that executes operations on machines which are remote from the Central server. You can use RAS operations to implement functionality interacting with systems throughout your network or over the Internet. For example, the use of RAS operations to integrate OO with other applications, platforms, and services.

Using a RAS operation instead of a scriptlet or command-line operation allows the operation to run hosted on a RAS. The advantage is that you can have multiple RASes running in different network segments and run operations on any of them.

RAS operations are written in either Java or .NET. The RAS operations for Java are packaged in JAR files and the ones for .NET are packaged in DLL files.

Note: RAS installed on a Windows server supports both Java and .NET RAS operations. However, RAS installed on a Linux server only supports Java RAS operations. It does not support .NET RAS operations.

About RAS

OO Central is installed with a default RAS named `RAS_Operator_Path`. You can also deploy OO RAS standalone, that is, on machines that are physically separate from the Central server. This process is explained in the *HP Operations Orchestration Installation and Upgrade Guide*.

The OO RAS contains the IAction interface, specifying how your action classes need to be built. For an operation to be accessible to the RAS for execution, the class that holds that operation needs to implement the IAction interface.

The OO server uses HTTP over the Secure Socket Layer (HTTPS) protocol to initiate communications between itself and the RAS, so you can deploy a RAS on the other side of a firewall or domain boundary and have it execute code for OO.

Currently the OO RAS supports the platform and language combinations shown in the following table:

Platform	Java	.NET
Windows 32-bit	X	X
Windows 64-bit	X	X
Linux 32-bit	X	
Linux 64-bit	X	

Creating IActions

To create an IAction, implement the IAction interface. This section explains how to use the IAction interface and provides guidelines and important points for creating IActions.

About the IAction interface

The IAction interface specifies how to build your Action classes. These classes define the RAS operations and are stored in the JAR or DLL files associated with the RAS.

The IAction interface uses the `execute` method, as well as methods that are specific to the Web application, standalone application, platform, or extension service for which you want the actions performed. The IAction interface mediates between OO and systems external to OO.

Syntax

Java

```
public interface IAction {

ActionTemplate getActionTemplate();

ActionResult execute(ISessionContext sessionContext,
ActionRequest actionRequest, IActionRegistry actionRegistry)
throws Exception;

}
```

.NET

```
public interface IAction

{

ActionTemplate GetActionTemplate();

ActionResult Execute(ActionRequest req, ISession s,
IActionRegistry reg);

}
```

Thus the IAction interface defines the following public methods needed to create IActions:

getActionTemplate

The `getActionTemplate` method returns the `ActionTemplate` object which describes the properties of the IAction to OO. These properties are shown in the following table:

Property	Description
The description of your operation.	Model the description after the operation descriptions included with the built-in OO RAS operations. The description should include the following: • An explanation of what the operation does. • Definitions of the operation inputs. • Definitions of the results returned by the operation. • Definitions of the responses returned by the operation.
A map of the inputs needed by the operation.	Set the key to the name of the input. You can leave the value as a blank string or set it to a `RASBinding` object. A `RASBinding` object has properties that correspond to most of the options available in the Studio **Inspector**'s **Inputs** tab. The order in which you add inputs to the map is the order in which they will appear in the operation once it is imported.
A map of the results returned by the IAction that are available to use in a flow.	Set the key to the name of the result. The value must be a blank string.
A map of the responses returned by the operation.	Set the key to the text that each response transition will show. The value must be the integer returned by the IAction's `returnCode`, telling Central which transition to follow when the operation completes.

Syntax

Java

```
public class ActionTemplate {
    private String description;
    private String overrideRas = "${overrideJRAS}";
    private Map parameters;
    private Map resultFields;
    private Map responses;

    public ActionTemplate() { }

    public ActionTemplate(String description, Map parameters,
    Map resultFields, Map responses) {

       this.description = description;
       this.parameters = parameters;
       this.resultFields = resultFields;
       this.responses = responses;

    }

    /**
     * Gets the description value for this ActionTemplate.
     * @return description
     */

    public String getDescription() {
        return description;
    }

    /**
     * Sets the description value for this ActionTemplate.
     * @param description
     */

    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * Gets the parameters value for this ActionTemplate.
     * @return parameters
     */

    public Map getParameters() {
        return parameters;
    }

    /**
     * Sets the parameters value for this ActionTemplate.
     * @param parameters
     */

    public void setParameters(Map parameters) {
        this.parameters = parameters;
    }

    /**
     * Gets the resultFields value for this ActionTemplate.
     * @return resultFields
     */

    public Map getResultFields() {
        return resultFields;
    }

    /**
     * Sets the resultFields value for this ActionTemplate.
     * @param resultFields
     */
```

```
    public void setResultFields(Map resultFields) {
        this.resultFields = resultFields;
    }

    /**
     * Gets the responses value for this ActionTemplate.
     * @return responses
     */

    public Map getResponses() {
        return responses;
    }

    /**
     * Sets the responses value for this ActionTemplate.
     * @param responses
     */

    public void setResponses(Map responses) {
        this.responses = responses;
    }

    /**
     * Sets the overrideRas value for this ActionTemplate.
     * @param overrideRas
     */

    public String getOverrideRas() {
    return overrideRas;
    }

    /**
     * Gets the overrideRas value for this ActionTemplate.
     * @param overrideRas
     */

    public void setOverrideRas(String overrideRas) {
    this.overrideRas = overrideRas;
    }
}
```

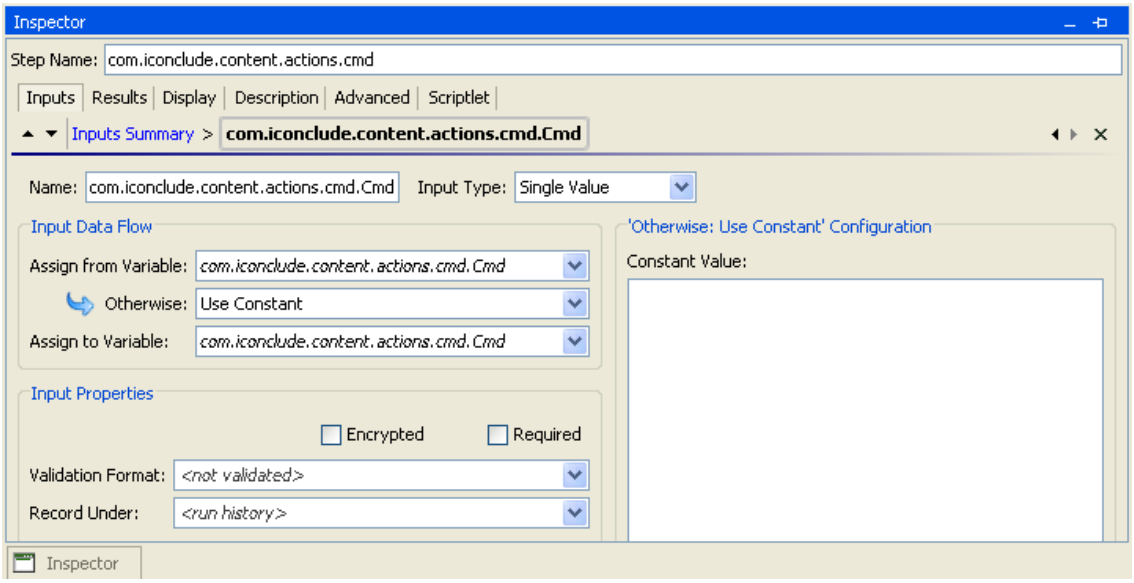
.NET

```
public class ActionTemplate {
    private String description;
    private String overrideRas = "${overrideJRAS}";
    private Map parameters;
    private Map resultFields;
    private Map responses;

    public ActionTemplate () { }

    public ActionTemplate (String description, Map parameters,
    Map resultFields, Map responses) {
       this.description = description;
       this.parameters = parameters;
       this.resultFields = resultFields;
       this.responses = responses;
    }

    /**
     * Gets the description value for this ActionTemplate
     * @return description
    */

    public String getDescription() {
        return description;
    }

    /**
     * Sets the description value for this ActionTemplate
     * @param description
     */

    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * Gets the parameters value for this ActionTemplate.
     * @return parameters
     */

    public Map getParameters() {
        return parameters;
    }

    /**
     * Sets the parameters value for this ActionTemplate.
     * @param parameters
     */

    public void setParameters(Map parameters) {
        this.parameters = parameters;
    }

    /**
     * Gets the resultFields value for this ActionTemplate.
     * @return resultFields
     */

    public Map getResultFields() {
        return resultFields;
    }

    /**
     * Sets the resultFields value for this ActionTemplate.
     * @param resultFields
     */
```

```
    public void setResultFields(Map resultFields) {
        this.resultFields = resultFields;
    }

   /**
     * Gets the responses value for this ActionTemplate.
     * @return responses
     */

    public Map getResponses() {
        return responses;
    }

    /**
     * Sets the responses value for this ActionTemplate
     * @param responses
     */

    public void setResponses(Map responses) {
        this.responses = responses;
    }

    /**
     * Sets the overrideRas value for this ActionTemplate.
     * @param overrideRas
     */

    public String getOverrideRas() {
    return overrideRas;
    }

    /**
     * Gets the overrideRas value for this ActionTemplate.
     * @param overrideRas
     */

    public void setOverrideRas(String overrideRas) {
    this.overrideRas = overrideRas;
   }
}
```

RASBinding objects

`RASBinding` objects expand the inputs in an `ActionTemplate` method. `RASBinding` allows you to identify exactly how the input is to be defined once it is imported in OO. This includes the default settings shown on the **Inputs** tab of the **Inspector** in Studio, as shown in the following figure:



The Inputs tab of the Studio Inspector

`RASBindings` have the properties shown in the following table:

Property	Description
encrypted (Boolean, false)	Determines whether the particular input should be encrypted.
required (Boolean, false)	Determines whether the particular input should be required.
assignTo (Boolean, true)	Determines whether the Assign to flow variable check box is checked.
assignFrom (Boolean, true)	Determines whether the Assign from flow variable check box is checked.
assignFromText (String, empty)	Determines the text in the Assign from flow variable field.
assignToText (String, empty)	Determines the text in the Assign to flow variable field.

Property	Description
type (INPUT_TYPE, INPUT_TYPE.Empty)	Determines the type of input: • `Empty` An empty binding that will become a prompt if not changed. • `Static` A static binding. Whatever is entered for the value property will become the value of this input. • `Prompt` A prompt-user binding. Whatever is entered in the `value` property will be the text that is used to prompt the user. • `value (String, empty)` The value that is assigned to either the `static`, or the `prompt user` field, depending on the specified type.

You can also use the `RASBindingFactory` method, whose main purpose is to quickly create `RASBindings` with default style behavior.

Syntax

Java

```
public class RASBindingFactory {

/*
 * Empty Bindings
 */

public static RASBinding createEmptyRASBinding(){
return new RASBinding();
}


public static RASBinding createEmptyRASBinding
    (boolean required, boolean encrypted)
{return updateBinding(createEmptyRASBinding(),
required, encrypted);
}

/*
 * Prompts
 */

public static RASBinding createPromptBinding(String value){
return createBinding(value, RASBinding.INPUT_TYPE.Prompt);
}

public static RASBinding createPromptBinding(String value,
 boolean required){
return updateBinding(createPromptBinding(value),
 required, false);
}

public static RASBinding createPromptBinding(String value,
 boolean required,boolean encrypted){
return updateBinding(createPromptBinding(value), required,
encrypted);
}

/*
 * Statics
 */

public static RASBinding createStaticBinding(String value){
return createBinding(value, RASBinding.INPUT_TYPE.Static);
}

public static RASBinding createStaticBinding(String value,
boolean required){
return updateBinding(createStaticBinding(value), required,
false);
}

public static RASBinding createStaticBinding(String value,
boolean required,boolean encrypted){
return updateBinding(createStaticBinding(value), required,
encrypted);
}

/*
 * Private section
 */

private static RASBinding createBinding(String value,
RASBinding.INPUT_TYPE type){
RASBinding r = new RASBinding();
```

Java

```
r.type = type;
r.value = value;
return r;
}

private static RASBinding updateBinding(RASBinding b,
boolean required,boolean encrypted){b.required=required;
b.encrypted=encrypted;
return b;
}

}
```

.NET

```
    public class RASBindingFactory
    {
        public static RASBinding createEmptyRASBinding()
        {
            return createGenericRASBindingWithValue(null,
                    RASBinding.BindingType.Empty);
      }
        public static RASBinding createEmptyRASBinding
                (Boolean required, Boolean encrypted)
        {
         return createGenericRASBindingWithValue(null,
          required, encrypted, RASBinding.BindingType.Empty);
        }
        public static RASBinding createPromptBinding(String value)
        {
            return
RASBindingFactory.createGenericRASBindingWithValue(value,
RASBinding.BindingType.Prompt);
        }
    public static RASBinding createPromptBinding(String value,
                Boolean required)
        {
   return RASBindingFactory.createGenericRASBindingWithValue(
            value, required, RASBinding.BindingType.Prompt);
        }
   public static RASBinding createPromptBinding(String value,
               Boolean required, Boolean encrypted)
        {
   return RASBindingFactory.createGenericRASBindingWithValue(
     value, required, encrypted, RASBinding.BindingType.Prompt);
       }
   public static RASBinding createStaticBinding(String value)
       {
   return RASBindingFactory.createGenericRASBindingWithValue(
                value, RASBinding.BindingType.Static);
       }
        public static RASBinding createStaticBinding
          (String value, Boolean required)
        {
       return RASBindingFactory.createGenericRASBindingWithValue(
              value, required, RASBinding.BindingType.Static);
       }
        public static RASBinding createStaticBinding(String value,
                   Boolean required, Boolean encrypted)
       {
       return RASBindingFactory.createGenericRASBindingWithValue(
         value, required, encrypted, RASBinding.BindingType.Static);
       }
        protected static RASBinding createGenericRASBindingWithValue(
            String value, RASBinding.BindingType type)
        {
            RASBinding r = new RASBinding();
```

.NET

```
            r.Binding = type;
            r.Value = (null != value ? value : "");
            return r;
}
        protected static RASBinding createGenericRASBindingWithValue(
         String value, Boolean required, RASBinding.BindingType type)
       {
            RASBinding r = new RASBinding();
            r.Required = required;
            r.Binding = type;
            r.Value = (null != value ? value : "");
            return r;
        }
        protected static RASBinding createGenericRASBindingWithValue(
        String value, Boolean required, Boolean encrypted,
        RASBinding.BindingType type)
        {
            RASBinding r = new RASBinding();
            r.Encrypted = encrypted;
            r.Required = required;
            r.Binding = type;
            r.Value = (null != value ? value : "");
            return r;
    }
```

execute

The `execute` method returns the `ActionResult` object from an IAction to Central after the code has been executed. The `ActionResult` object can return the results shown in the following table:

Result	Description
exception (String, empty)	This result should be the stack trace for any exception that your operation encounters. This result is most often used inside of the try/catch block for the operation. It should be left blank if no exception is encountered.
returnCode (int, 0)	This result indicates which transition Central should follow after the operation completed. It should be set to values that can be mapped to the responses in the `ActionTemplate` for this operation. See "getActionTemplate" on page 28 for more information.
Results	This is a map of the results sent back to central. These results are available on the **Results** tab in Studio. They can be filtered or used as flow variables or flow results by other operations or flows.

The `ActionResult` object extends the `Map` object, so any other results that you have defined in the `ActionTemplate` can also be returned to Central. The key field is the result name and the value field is a string value.

Syntax

Java

```
public class ActionResult extends Map {
  private String exception;
    private int returnCode;
    private String sessionId;

    public ActionResult() {
    }

    public ActionResult(String exception, MapEntry[] entries,
           int returnCode, String sessionId) {
    super(entries);
      this.exception = exception;
      this.returnCode = returnCode;
      this.sessionId = sessionId;
    }

    /**
     * Gets the exception value for this ActionResult.
     * @return exception
     */

    public String getException() {
        return exception;
    }

    /**
     * Sets the exception value for this ActionResult.
     * @param exception
     */

    public void setException(String exception) {
        this.exception = exception;
    }

    /**
     * Gets the returnCode value for this ActionResult.
     * @return returnCode
     */

    public int getReturnCode() {
        return returnCode;
    }

    /**
     * Sets the returnCode value for this ActionResult.
     * @param returnCode
     */

    public void setReturnCode(int returnCode) {
        this.returnCode = returnCode;
    }

    /**
     * Gets the sessionId value for this ActionResult.
     * @return sessionId
     */

    public String getSessionId() {
        return sessionId;
    }

    /**
     * Sets the sessionId value for this ActionResult.
     * @param sessionId
     */

    public void setSessionId(String sessionId) {
```

```
        this.sessionId = sessionId;
    }
}
}
```

.NET

```
public class ActionResult : Map

{
public int code = (int)ResultCode.SUCCESS;
public string sessionId = "new session";
public string exception;
public ActionResult () {}

public int GetReturnCode() { return code; }
public void SetReturnCode(int c) { code = c; }
public string GetException() { return exception; }
public void SetExceptionstring ex) { exception = ex; }
public string GetSessionId() { return sessionId; }
public void SetSessionId(string sid) { sessionId = sid; }

}
```

Guidelines for creating IActions

When you create a Java or .NET IAction, use the following guidelines:

- Create any static constants you need.
- Define your `ActionTemplate`. This defines the inputs, outputs, and responses for the IAction. If an input, output, or response does not appear when you open the IAction in Studio, it means that you have made an error in the `ActionTemplate`.
- Add an `execute` method and make sure that it uses a `try/catch` block.
- Convert the inputs from the `ActionTemplate` to variables in your code and run the code.
- Make sure that every result has a defined value, even if that value is an empty string. This is necessary in case an exception is thrown.
- Set a value for the return code. The response (return code) for success is always `0` and for failure it is always `-1`.
- When possible, write a meaningful error message in case your operation fails.
- Only the inputs, responses, and results defined in the `ActionTemplate` are created automatically. See "getActionTemplate" on page 28 for more information.
- Make sure to handle system accounts properly. For Java IActions, use the `StringUtils.resolveString` method. For .NET IActions, use the `Identities` methods.

Important notes for creating Java IActions

When creating a Java IActions:

- Check the inputs to make sure that they have no `null` values. If you use the `com.opsware.pas.content.commons.util.StringUtils.resolveString method`, `null` inputs are automatically converted into empty strings. See "Useful Java Commons Library classes" on page 56for more information. Optional inputs can be strings of zero length.

- Use the `StringUtils.resolveString` method, as it will handle system accounts for you. See "Useful Java Commons Library classes" on page 56 for more information.
- Any results you want the user to have access to must be included in the `ActionTemplate`.
- Do not use an instance variable in an IAction if it is unique to a given run of the IAction.
- Do not write to an instance variable.

Important notes for creating .NET IActions

When creating a .NET IAction:

- Use the .NET `Convert.ToString()` method for handling all inputs except system accounts.
- Handle system accounts by using the `Identities` methods in **Commons.dll**. See "Identities class" on page 56 for more information.
- Any results you want the user to have access to must be included in the `ActionTemplate`.
- Do not use an instance variable in the IAction if it is unique to a given run of the IAction.
- Do not write to an instance variable.

Implementing Java IActions

Java IActions are Java classes that can be imported and used by OO. This section explains:

- "Required development files for Java IActions" below.
- "Loading Java IActions into Studio" on next page.
- "Using third-party libraries for Java IActions" on next page.
- "Debugging Java IActions" on page 49.

This section also contains a complete example of the code needed to create a Java IAction. See "Java IAction code example" on page 51.

Required development files for Java IActions

The following files are required for developing Java IActions:

- JRAS-sdk-9.00.jar
- ContentCommons-9.00.jar

These files are included in the OO SDK home directory, in the **lib** folder.

Loading Java IActions into Studio

To import Java IActions into Studio:

1. Create a JAR file with your IAction classes in it. You can create more than one IAction JAR files.
2. Stop the **RSJRAS** service. This is the Windows service running the RAS.
3. Copy your JAR file to the **RAS\Java\Default\repository** folder in the OO home directory.
4. Copy any additional libraries that your IActions use to the **RAS\Java\Default\repository\lib\<jarName>** folder. **<jarName>** is the name of the J file, without the **.jar** extension, found in the OO home directory.

   For more information on IAction JAR files and third-party libraries, see "Using third-party libraries for Java IActions" below.
5. Restart the **RSJRAS** service.
6. In Studio, create a folder for the IActions.
7. Select the folder and then, from the **File** menu, click **Create Operations from RAS**.
8. In the RAS import dialog box, select **RAS_Operator_Path**, assuming you put the JAR file into the default RAS, and then click **OK**. The IActions are imported.

Note: The code in one IAction JAR file cannot use the code in another IAction JAR file. Reimporting the IAction JAR file will generate different **Universally Unique Identifiers** (UUIDs) for the operations.

Using third-party libraries for Java IActions

You can have multiple IAction JAR files, each referring different versions of third-party libraries. The **JRAS** service looks for third-party Java libraries under the OO home directory, in the following folders:

- RAS\Java\Default\repository\lib\<jarName>
- RAS\Java\Default\repository\lib
- RAS\Java\Default\webapp\WEB_INF\lib

<jarName> is the name of the JAR file, without the **.jar** extension, found in the OO home directory.

In addition to these paths, if an IAction JAR file has a main manifest attribute named `Custom-Libraries`, the value of this attribute will be processed as a comma-delimited list of additional folders to load. These folders are relative to the **RAS\Java\Default\repository\lib** folder. Libraries referenced this way will be loaded between the **RAS\Java\Default\repository\lib\<jarName>** and **RAS\Java\Default\repository\lib** folders.

Important: The **RAS\Java\Default\repository\lib** folder is shared by all IAction JAR files. Do not put your own third-party libraries in this folder. Adding libraries to this folder can the break out-of-the-box content.

The **RAS\Java\Default\webapp\WEB_INF\lib** folder is intended to be used by the **RAS** service itself. IAction libraries should not have their own third-party libraries put in this folder. Adding libraries to this folder can break the out-of-the-box content, or the RAS service.

The preferred method is that each IAction JAR file author adds a folder named **RAS\Java\Default\repository\lib\<jarName>**, where **<jarName>** is the name of the IAction JAR file, without the **.jar** extension. Third party libraries should go into this folder.

For example, if you have an integration called **TestApp**, your IAction will be in a JAR file called **TestApp.jar**. Assume that you need the integration library **TestLib.jar** and the third party libraries **TestThirdPartyLib1.jar** and **TestThirdPartyLib2.jar** for **TestApp** to run.

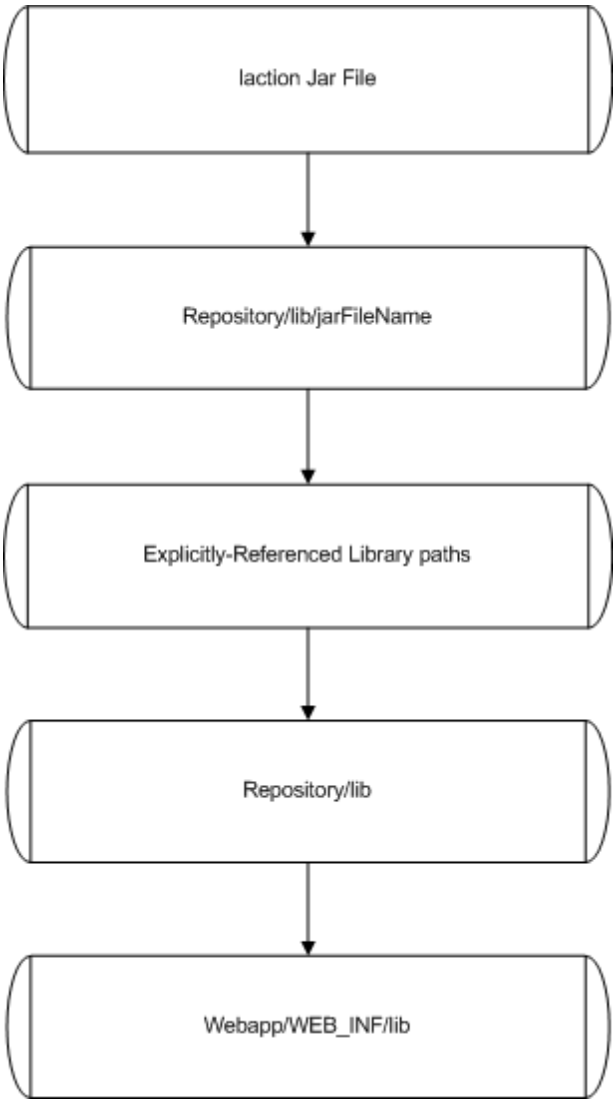
The layout of the **TestApp** integration JAR files is:

- RAS\Java\Default\repository\lib\TestApp\TestLib.jar
- RAS\Java\Default\repository\lib\TestApp\TestThirdPartyLib1.jar
- RAS\Java\Default\repository\lib\TestApp\TestThirdPartyLib2.jar
- RAS\Java\Default\repository\TestApp.jar

If multiple IAction JAR files need to share a custom set of libraries, you can make one or more specifically named folders in the **RAS\Java\Default\repository\lib** folder, and explicitly reference them through the `Custom-Libraries` manifest attribute. However, you should do this only if you have no other choice. Use the preferred method mentioned above, so your integration libraries are grouped in one place.

For example, for **TestApp IActions** you also want to use different sets of libraries, **AnotherSetLib1.jar** and **AnotherSetLib2.jar**. You must put theses libraries in different directory. Create a directory called **RAS\Java\Default\repository\lib\TestApp2** and put **AnotherSetLib1.jar** and **AnotherSetLib2.jar** in it. The **TestApp.jar** manifest file must include the `Custom-Libraries` to point to **RAS\Java\Default\repository\lib\TestApp2\AnotherSetLib1.jar** and **AnotherSetLib2.jar**.

`Custom-Libraries`: **RAS\Java\Default\repository\lib\TestApp2\AnotherSetLib1.jar**, **RAS\Java\Default\repository\lib\TestApp2\AnotherSetLib2.jar**



How RAS resolves third-party Java libraries at run time

Debugging Java IActions

To enable remote RAS debugging for Java IActions:

1. Stop the **RSJRAS** service.
2. Copy your JAR file to the **RAS\Java\Default\repository** folder in the OO home directory.
3. Copy any additional libraries you might be using for the IActions to the **RAS\Java\Default\repository\lib** folder in the OO home directory.
4. Open the **RAS\Java\Default\webapp\conf\wrapper.conf** file in the OO home directory, using your preferred text editor.
5. Uncomment the following debug line:

   ```
   (#wrapper.java.additional.2=-Xdebug -Xnoagent -Djava.compiler=NONE
   -Xrunjdwp:transport=dt_socket,address=8070,server=y,suspend=y
   ```

   This suspends the **RSJRAS** service startup until a remote debugger is configured to use port `8070`.
6. Change the following debug line:

   ```
   wrapper.java.additional.2
   ```

   to

   ```
   wrapper.java.additional.3
   ```

   or

   ```
   wrapper.java.additional.n
   ```

   where `n` is the last number you used, incremented by one.
7. Restart the **RSJRAS** service.
8. Configure your remote debugger to use the port specified in the **wrapper.conf** file. The default port is `8070`, but you can change it to any unused port.
9. Set a breakpoint in your Java source code where you want to stop and connect to the remote debug session listening on the port specified in step 8.
10. Execute a flow that uses your operation.
11. Debug the IAction code.

To disable remote RAS debugging for Java IActions:

1. Stop the **RSJRAS** service.
2. Open the **RAS\Java\Default\webapp\conf\wrapper.conf** file in the OO home directory.
3. Comment out the following debug line:

   ```
   (#wrapper.java.additional.2=-Xdebug -Xnoagent -Djava.compiler=NONE
   -Xrunjdwp:transport=dt_socket,address=8070,server=y,suspend=y)
   ```
4. Change the following debug line:

`wrapper.java.additional.2`

to

`wrapper.java.additional.3`

or

`wrapper.java.additional.n`

where `n` is the last number you used, incremented by one.

5. Start the **RSJRAS** service.

Java IAction code example

The following is an example of a Java IAction that reads a file and returns the contents:

```
public class ReadFile implements IAction {

   private static final String RETURNRESULT = "returnResult";
   public static final int PASSED = 0;
   public static final int FAILED = 1;

   @Override
   public ActionTemplate getActionTemplate() {
      ActionTemplate actionTemplate = new ActionTemplate();
      actionTemplate.setDescription(ReadFile.DESCRIPTION);

      RASBinding arg1 = RASBindingFactory.createPromptBinding
       ("Source File:", true);

      Map parameters = new Map();
      parameters.add("source", arg1);
      actionTemplate.setParameters(parameters);

      Map resultFields = new Map();
      resultFields.add("fileContents", "");
      resultFields.add(RETURNRESULT, "");
      actionTemplate.setResultFields(resultFields);

      Map responses = new Map();
      responses.add("success", String.valueOf(PASSED));
      responses.add("failure", String.valueOf(FAILED));

      actionTemplate.setResponses(responses);
      return actionTemplate;
   }

   @Override
   public ActionResult execute(ISessionContext session,
   ActionRequest request, IActionRegistry registry)
   throws Exception {

    ActionResult result = new ActionResult();
    String separator = (System.getProperty("line.separator")
         != null)? System.getProperty("line.separator") : "\n" ;

      String line = null;
      File file = null;
      FileReader fReader = null;
      BufferedReader bReader = null;

      try {
         file = new File ActionRequestUtils.resolveStringParam(
            request, "source"));

         StringBuilder fileContents = new StringBuilder();
         fReader = new FileReader(file);
         bReader = new BufferedReader(fReader);
         while ((line = bReader.readLine()) != null) {
            fileContents.append(line);
            fileContents.append(separator);
         }
```

```
            result.add("fileContents", fileContents.toString());
            result.add(RETURNRESULT, "successfully read file");
            result.setReturnCode(PASSED);
        } catch (Exception e) {
            result.setReturnCode(FAILED);
            result.setException(StringUtils.toString(e));
            result.add(RETURNRESULT, e.getMessage());

        } finally {
            if (bReader != null)
               bReader.close();
            if (fReader != null)
              fReader.close();
        }

        return result;
    }

    private static String DESCRIPTION = ""
        +"<pre>Reads the contents of a file and returns it\n"
        +"Inputs:\n"
        +"source - path to file to read\n"
        +"\n"
        +"Responses:\n"
        +"success - successfully read file"
        +"failure - failed to read the file\n"
        +"\n"
        +"Extra Results:\n"
        +"fileContents - the contents of the file\n\n</pre>";
}
```

Implementing .NET IActions

.NET IActions are .NET assemblies that can be imported into and used by OO. This section explains:

- "Required development files for .NET IActions" below.
- "Loading .NET IActions into Studio" on next page.
- "Debugging .NET IActions" on next page.

This section also contains a complete example of the code needed to create a .NET IAction. See ".NET IAction code example" on next page.

Required development files for .NET IActions

To develop .NET IActions, you need:

- IAction.dll
- RCAgentLib.dll
- Commons.dll

These files are included in the OO home directory, in the **RAS\Java\Default\repository** folder.

Loading .NET IActions into Studio

To import .NET IActions into Studio:

1. Create a DLL file with your IAction classes in it.
2. Stop the **RSJRAS** service. This is the Windows service that runs the RAS.
3. Copy your DLL file to the **RAS\Java\Default\repository** folder in the OO home directory.
4. Copy any additional DLL libraries you might be using for the IActions to the same folder.
5. Restart the **RSJRAS** service.
6. In Studio, create a folder for the IActions.
7. Select the folder and then, from the **File** menu, click **Create Operations from RAS**.
8. In the **RAS import** dialog box, select **RAS_Operator_Path**, assuming that you put the DLL file into the default RAS, and then click **OK.** The IActions are imported.

Debugging .NET IActions

To enable remote RAS debugging for .NET IActions:

1. Stop the **RSJRAS** service.
2. Copy your DLL and PDB files to the **RAS\Java\Default\repository** folder in the OO home directory. PDBs are .NET debug files.
3. Copy any additional libraries you might be using for the IActions to the same folder.
4. Restart the **RSJRAS** service.
5. Configure your debugger to use the **java.exe** process that hosts the **RSJRAS** service.
6. Set a breakpoint in your .NET source code where you want to stop.
7. Run a flow that uses your operation.
8. Debug the IAction code.

To disable remote RAS debugging for .NET IActions:

1. Disconnect the debugger from the **java.exe** process.
2. Stop the **RSJRAS** service.
3. Remove the PDB files.
4. Restart the **RSJRAS** service.

.NET IAction code example

The following is an example of a .NET IAction that reads a file and returns the contents:

```
using System;
using System.IO;
using System.Text;
using System.Collections;
```

```
using System.Diagnostics;
using System.Globalization;
using com.iconclude.agent;
using System.Text.RegularExpressions;

using DiskServices;
using dotNET_Commons;

namespace com.hp.oo.content.sdk
{
    public class ReadFile : IAction
    {
        public ActionResult Execute(ActionRequest request,
      ISession session, IActionRegistry registry)
        {
            ActionResult result = new ActionResult();
            StreamReader sReader = null;
            String line = null;

            try
            {

                string strSource = Convert.ToString(
                   request.parameters["source"]);

                StringBuilder fileContents = new StringBuilder();

                Identities.ChangeUserContext(request);

                sReader = File.OpenText(strSource);

                while ((line = sReader.ReadLine()) != null)
                  fileContents.AppendLine(line);
            }
            catch (Exception e)
            {
                ret.SetReturnCode(ReturnCodes.FAILED,
                    e.Message);
                ret.SetException(e.ToString());
            }
            finally
            {
                Identities.UnchangeUserContext(req);
            }

            return ret.GetActionResult();

        }

    public ActionTemplate GetActionTemplate()
        {
            ActionTemplateEx template = new ActionTemplateEx();
              template.SetDescription("WriteToFile");


            RASBinding arg1 = RASBindingFactory.createPromptBinding(
                            "FileName:");

            arg1.AssignFrom(true);
            arg1.AssignTo(true);
```

```
            RASBinding arg2 = RASBindingFactory.createPromptBinding(
                               "Text To Write:");

            arg2.AssignFrom(true);
            arg2.AssignTo(true);

            RASBinding arg3 = RASBindingFactory.createPromptBinding(
                              "Alternate Credentials - UserName:",
                                false, false);

            arg3.AssignFrom(true);
            arg3.AssignTo(true);

            RASBinding arg4 = RASBindingFactory.createPromptBinding(
               "Alternate Credentials - Password:", false, true);

            arg4.AssignFrom(true);
            arg4.AssignTo(true);

            template.AddParameter("File", arg1);
            template.AddParameter("Contents", arg2);
            template.AddParameter("user", arg3);
            template.AddParameter("password", arg4);

            template.AddResponse("success", (int)ReturnCodes.PASSED);
            template.AddResponse("failure", (int)ReturnCodes.FAILED);

            return template.GetActionTemplate();
        }
    }
}
```

Useful Java Commons Library classes

The **com.opsware.pas.content.commons.util StringUtils** class is available for general use and may be helpful as you develop content. It is located in the **ContentCommons.jar** file in the **RAS\Java\Default\repository\lib** folder in the OO home directory. This class is maintained to ensure backward compatibility.

com.opsware.pas.content.commons.util StringUtils class

This is a helper class in Java for handling system account inputs and inputs that are null or missing.

Properties

Name	Description
None	

Constructors

Name	Description
None	All methods are static.

Methods (all are static)

Name	Description
isNull(String)	Boolean specifying whether the value passed is null.
resolveString (ActionRequest, String)	Resolves the value of `String` from the input map in `ActionRequest`.

Useful .NET Commons Library classes

The classes described in this section are available for general use and might be helpful as you develop content. They are located in the **Commons.dll** file, in the **RAS\Java\Default\repository** folder in the OO home directory. These classes are maintained to ensure backward compatibility.

Identities class

The `Identities` class allows you to deal with user permissions (system accounts) and to perform user impersonation for several operations.

There are two impersonation styles:

- The one that is attempted first is an **Inter-Process Communication** (IPC) connection to the remote machine.

- If the previous style fails, or if the authentication is performed against the local machine instead of the RAS, then local thread impersonation is attempted.

In most cases, the `Identities` class uses an `ActionRequest` method, and handles system accounts.

Properties

Name	Description
DestinationHost	(Internal) The machine host value when dealing with copying files.
File	(Internal) The file path for impersonation purposes.
Host	The remote host to connect to.
Password	The password to use for the connection.
SourceHost	(Internal) The machine host for dealing with copying files.
UserName	The username to use for the connection.

Constructors

Name	Description
Identities()	Default.

Methods

Name	Description
ChangeUserContext (ActionRequest)	Assuming that `ActionRequest` has the proper inputs defined, this method impersonates the user. If you use this method, you must use the `UnchangeUserContext(ActionRequest)` method when the impersonation is finished.
ChangeUserContext (string host, string user, string password)	Attempts to establish the connection and impersonate the specified user. If you use this method, you need to use the `UnchangeUserContext(string)` when the impersonation is finished.
GetUserPass (ActionRequest, string username key, string password key)	This is a useful method if you have inputs that are system accounts, but are not valid input names. `ActionRequest` must contain the `username key` and `password key` inputs, passed in to specify which input to use in order to pull the actual user name and password values. These values can then be read back using the `UserName` and `Password` properties.
GetUserPass (ActionRequest)	This method is the same as `GetUserPass(ActionRequest, string username key, string password key)`, except that the `username key` and `password key` inputs are passed automatically.

Name	Description
UnchangeUserContext (ActionRequest)	If you used the `ChangeUserContext(ActionRequest)` method, use this method with the `ActionRequest` used during the call in order to reverse the impersonation.
UnchangeUserContext (string hostname)	If you used the `ChangeUserContext(string)` method, use this method with the host name used during the call to reverse the impersonation.

If you use the `ChangeUserContext(ActionRequest)` method, the names of the inputs mapped in the `ActionRequest` have to conform to the inputs shown in the following table. The inputs are case sensitive.

Input Type	Valid Input Names
Host	• host • hostname
Username	• username • user • User • F5Username • altuser
Password	• password • Password • altpass • pass • F5Password • Pass

If the inputs do not comply to these specifications, you need to use the `ChangeUserContext (string host, string user, string password)` impersonation method.

Important: If you use any of the impersonation functions, make sure you have the unimpersonation area wrapped in a finally block. That way you can always roll back your impersonation.

Remarks

The most flexible and secure way to communicate between Windows servers is by making authenticated IPC connections. This can be done on the command line, using net `\\machinename\ipc$` and supplying the necessary credentials. This is how the `Identities` class does it as well. However, only one IPC connection can exist between any two machines.

Therefore, only one IPC connection can exist between the RAS and a given server at any given time. Because of this, the RAS is designed as follows:

1. The RAS receives a request to impersonate a user. To check the request in order to see whether a remote machine is requested, the RAS examines the hostname, the fully qualified domain name (FQDN), and all of the IP addresses registered on the machine. If the requested machine is not remote, the RAS attempts thread elevation. If the requested machine is remote, the RAS continues to the next step.
2. The RAS searches the hostname to check whether an IPC connection with the same user name and password is currently mapped. If it is, the RAS adds a flag to indicate that another operation is using the same connection, and the impersonation class returns.

   If a connection doesn't exist:

   - A new request to map the remote machine's IPC share is sent using the standard Windows API.
   - The RAS adds the first flag for this machine.
   - The impersonation class returns.
3. Once the operation completed and called the unimpersonation method in this class, the RAS checks to see whether it used a thread impersonation or an IPC connection.
   - If the RAS used a thread impersonation, it restores the thread to its old credential set.
   - If the RAS used an IPC connection, the connection use count is decremented. If the use count is now at zero, the IPC connection with the remote machine is closed and the class returns. If the use count is not zero, the class simply returns.

Password class

This class generates random passwords.

Properties

Name	Description
None	

Constructors

Name	Description
None	

Methods

Name	Description
static GenerateRandom(int length, int numberOfNonAlphaNumericCharacters)	A static method to generate random passwords.

Chapter 3

Finding and running flows from outside Central

In most cases, you use Central to run the flows you created in Studio. However, there are situations when you want to find or run flows without using Central. For instance, you want to run a flow from an external application, such as **Microsoft System Center Operations Manager**, from a script, or batch file.

Ways to find and run flows from outside Central

- "Creating a URL for running a flow" on next page. Run a **Guided Run** or **Run All** flow from a Web browser.
- "Running a flow from a command line" on next page. Or from an application that can use a command line.
- "Finding and running flows with tools that access the REST service" on page 64. Run flows using the Internet.
- "Accessing the WSCentralService WSDL" on page 73. Access Central features programmatically through the WSCentralService WSDL SOAP API.

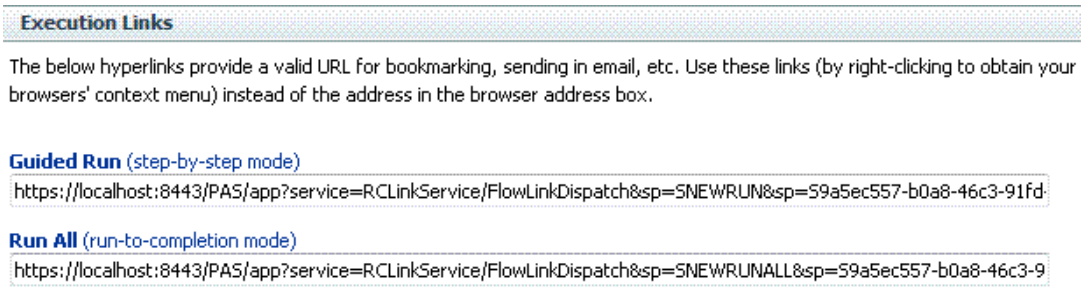
Important: Although you do not use it for managing flows externally, Central must be running when you do so.

Running flows with URLs created in Central

In Central, you can obtain a valid URL and use it to run a **Guided Run** or **Run All** flow from a Web browser.

To use a URL created in Central:

1. In Central, click the **Flow Library** tab, navigate to the flow, and then click the flow name to open the flow preview.
2. In the left pane, under **Execution Links,** select and copy the URL in the box below the desired type of run: **Guided Run** or **Run All**.



3. Open a new browser window and paste the URL in the address box.
4. If the flow has required inputs, modify the URL by adding input parameters and input values.

   For more information, see "Specifying the inputs for a flow in a URL" on next page.

Note: You can also paste the URL in a document or e-mail, but if you paste it into an e-mail, you cannot pass input parameters in the URL.

Running a flow from a command line

You can run a flow from a command line using a correctly formatted URL.

For information on creating the flow's URL, see "Creating a URL for running a flow" below.

When running a flow from a command line, use the following guidelines:

- The flow must to be self-contained. This means that the flow does not contain user prompts.
- You can pass input parameters and values to the flow, in the URL.
- If an input requires a flow variable, it is not compulsory that you define the variable when you create the flow in Studio. You can create and pass the flow variable to the input, using an input parameter in the URL.

Creating a URL for running a flow

The format for a URL that runs a flow from a command line is :

```
https://<hostname>:<port>/<path>/<flow>
```

When creating a URL, be aware that:

- There are two ways to identify the flow: by name or by the **Universally Unique Identifier** (UUID). For information on how to identify the flow in the URL, see "Identifying the flow in the URL" on next page.
- The initial inputs (flow input values) required for the flow to run must be included in the URL. For information on specifying initial input values, see "Specifying the inputs for a flow in a URL" on next page.
- You can modify the URL to provide a result from the flow asynchronously, without waiting for the flow to complete its run. For aditional information, see "Running flows asynchronously using a URL" on page 63.

Identifying the flow in the URL

In a URL that runs a flow, you can identify the flow by:

- The name of the flow. The following URL identifies the flow by its name, **TestFlow**:

  `https://localhost:8443/PAS/services/rest/run/Library/MyFolder/TestFlow`

- The flow's UUID. The following URL identifies the flow by its UUID (`503c2500-7aae-11dd-a3b5-0002a5d5c51b`):

  `https://localhost:8443/PAS/services/rest/run/503c2500-7aae-11dd-a3b5-0002a5d5c51b`

Specifying the inputs for a flow in a URL

You can specify the inputs for a flow by using input parameters, also known as **init params**, in the URL. This allows you to run the flow without any user interaction.

Init params are separated from the flow identifier by a question mark (`?`). Each init param takes the form `name=value`. If you use more than one init param, separate the init params with an ampersand (`&`).

Example

The URL in this example runs the **MyFlow** flow, passing the input parameter `name0` with a value of `val0` and the init param `input1` with a value of `yes`.

`https://localhost:8443/PAS/services/rest/run/Library/MyFolder/MyFlow?name0=val0?input1=yes`

When naming init params:

- Do not name init params `service` or `sp`, as these are reserved names.
- If a flow uses one of the reserved names for an input, protect your flows from errors that can result from using these reserved names, by defining a prefix for all init param names used in the URL. As long as a required init param prefix is specified, you must use it for all the init params for flows started by means of a URL in Central, including those that do not use the reserved names.

To define a prefix for init param names

1. Log on to Central with an account that has OO administrative rights.
2. On the **Administration** tab, click the **System Configuration** tab.
3. In the **General Settings** area, in the **Value** box of the **Prefix for init params for flow invocation through URLs using the GUI** row, type the prefix that you want to use for the input parameters in a URL.
4. Click **Save General Settings**.
5. Restart Central.

When defining a prefix for input parameters, avoid using the types of characters shown in the following table:

Types of characters to avoid using	Examples
Characters that are reserved in URLs.	; / ? : @ = &
Characters that can be misunderstood in URLs.	{ } | \ ^ ~ [ ] `

If the flow you are starting has a multi-instance step, or has an input whose value is a list of values, use the separator character defined by the flow's author in Studio to separate the values for the input that has multiple values. The default separator is a comma. Also do this if your flow has an input that is a list of values.

The following sample URL shows how to use a prefix for initial inputs in a URL and how to specify a list of values for a single input:

```
https://localhost:8443/PAS/services/rest/run/Library/MyFolder/TestFlow?_xx_input1=10.0.0.100,
10.0.0.101&_xx_input2=8443&_xx_input2=8443
```

In this example:

- The prefix for the init params has been defined as `_xx`.
- The flow specified in the URL has a multi-instance step with the separator character defined as an ampersand (`&`).
- The init param `_xx_input1` has two values—the IP addresses `10.0.0.100` and `10.0.0.101`, using the default value-list separator character, comma [`,`].

Notes:

- You only have to specify values for inputs that get their values from user prompts or that have not been assigned a value or a way to get a value.
- You do not have to specify a value for an input that has a specific value assigned to it, or a set of values, as in multi-instance steps or steps that get their values from an `Iterator` operation.
- You do not have to specify a value for an input that gets its value from a system account or from the logged-in user credentials.

Running flows asynchronously using a URL

You can obtain a result from a flow without waiting for it to finish, by running the flow asynchronously. This can be very useful if you run flows with multi-instance steps or multiple input values.

To run a flow asynchronously, in the URL that runs the flow, replace `/run/` with `/run_async/`.

For example, the following URL runs the **Connectivity Test** flow asynchronously, using the `run_async` parameter. This flow has a multi-instance step with multiple values for the target input.

```
https://localhost:8443/PAS/services/http/run_async/Library/MyFolder/Connectivity
Test?&host=localhost&target=55.55.0.47,55.55.0.49
```

Finding and running flows with tools that access the REST service

The **Representational State Transfer** (REST) is an architectural style that uses existing Internet technologies and protocols such as HTTP and XML. This section describes two command-line tools that access the **REST** service, allowing you to find and run flows using the Internet:

- The **Wget** tool allows you to send a user name and a password in the command line, but does not provide feedback as to whether your call worked correctly. See "Running flows using Wget" below.
- With **RSFlowInvoke.exe** or **JRSFlowInvoke.jar** you can send encrypted passwords over the Internet. In addition, **RSFlowInvoke** and **JRSFlowInvoke** return XML or HTML feedback, verifying whether your call worked. See "Finding and running flows using RSFlowInvoke or JRSFlowInvoke" on page 66.

Important: Wget, **RSFlowInvoke.exe**, and **JRSFlowInvoke.jar** can take command-line parameters or URLs to specify the flows you want to manage with them. If you use a URL, you need to enclose it with quotation marks.

Running flows using Wget

Wget is a command-line tool you can use to download and run flows from the Internet. You can download **Wget** from the *GNU Wget Web page*.

The basic **Wget** syntax is: `wget {<options>} {<URL>}`

Wget downloads and runs flows specified in the URL that is contained on the command line. It can use the HTTP, HTTPS, and FTP protocols. The **Wget** options are explained in the *GNU Wget Manual*.

Important: Enclose the URL on the command line with quotation marks.

The following examples show how to download and run flows using a URL in a **Wget** command line:

Example 1

To download the flows in the **MyFolder** folder, you use the command:

```
wget --http-user=rsadmin --http-
password=iconclude"https://localhost:8443/PAS/services/rest/list/MyFolder/"
```

- The **Wget** `-O` option specifies that all error messages are logged to the default log file **wget.log**.
- The **Wget** `http-user=user` and `http-password=password` options specify a `rsadmin` user name and an `iconclude` password for the HTTP server.

Example 2

This example uses the **Wget** `no-check-certificate` option to skip **Secure Sockets Layer** (SSL) checking.

```
wget --no-check-certificate --http-user=rsadmin --http-password=iconclude
"https://localhost:8443/PAS/services/rest/list/MyFolder/"
```

Example 3

This example runs the flow specified in the URL and passes the `name0` input variable with a `val0` value.

```
wget --http-user=rsadmin --http-password=iconclude
"https://localhost:8443/PAS/services/rest/run/Library/MyFolder/MyFlow?name0=val0"
```

The next examples works with an XML file that has the following basic layout.

XML file layout

```
<?xml version="1.0"?>
<run>
<request>
    <arg name="name1">value1</arg>
    <arg name="name2">value2</arg>
</request>
</run>
```

Example 4

This example uses `POST` as the method to run an XML-encoded flow from the **C:\run-config.xml** file.

```
wget --http-user=rsadmin --http-password=iconclude --post-file="C:\run-config.xml"
--header "Content-Type: text/xml"
"https://localhost:8443/PAS/services/rest/run/Library/MyFolder/MyFlow"
```

Example 5

This example runs an XML-encoded flow from a command line using the `post-data=string` **Wget** option.

```
wget --http-user=rsadmin --http-password=iconclude --post-data=
"<?xml version=\"1.0\" ?><run><request>
<arg name=\"name0\">value0</arg><arg name=\"name1\">value1</arg>
<arg name=\"name2\">value2</arg></request></run>"
--header "Content-Type: text/xml"
"https://localhost:8443/PAS/services/rest/run/Library/MyFolder/MyFlow"
```

The following example shows the returned XML format:

```
<?xml version="1.0" encoding="UTF-8"?>
<runResponse>
    <runReturn>

<item>
    <name>runId</name>
    <value>23</value>
</item>

<item>
    <name>runReportUrl</name>
        <value>https://localhost:8443/PAS/app?service=RCLinkService/ReportLinkDispatch
         &amp;sp=SINDIVIDUAL_REPAIR_LEVEL&amp;sp=
         Sc2bcb72f-6d6b-4a2d-a678-de21a1feac81&amp;sp=l0&amp;sp=l23</value>
</item>

<item>
    <name>runStartTime</name>
    <value>09/17/08 13:00:54</value>
</item>

<item>
    <name>runEndTime</name>
    <value>09/17/08 13:00:54</value>
</item>

<item>
    <name>runHistoryId</name>
    <value>23</value>
</item>

<item>
    <name>flowResponse</name>
    <value>success</value>
</item>

<item>
    <name>flowResult</name>
    <value>{Field 1=value0;Field 2=value1;FailureMessage=;TimedOut=;Result=;}</value>
</item>

<item>
    <name>flowReturnCode</name>
    <value>Resolved</value>
</item>

    </runReturn>
```

Finding and running flows using RSFlowInvoke or JRSFlowInvoke

RSFlowInvoke.exe, or the Java version, **JRSFlowInvoke.jar**, is a command-line utility that you can use to list, run, and search for flows outside Central.

The following sections describe how to use **RSFlowInvoke** and **JRSFlowInvoke** in a number of ways:

- "Using RSFlowInvoke or JRSFlowInvoke from a command line" on page 69.
- "Using RSFlowInvoke or JRSFlowInvoke in a script or batch file" on page 69.
- "Searching for a flow using JRSFlowInvoke.jar" on page 70.
- "Creating an encrypted password" on page 71.

You can run **RSFlowInvoke** or **JRSFlowInvoke** on any machine from which you can log on to Central, using HTTPS to the default port `8443`. This makes **RSFlowInvoke** and **JRSFlowInvoke** useful for starting a flow from an external system or application that can use a command. For example, a monitoring program such as Microsoft System Center Operations Manager.

To use **RSFlowInvoke**, you need to register it with the **.NET Global Assembly Cache**. See "Registering RSFlowInvoke with the Global Assembly Cache" on page 72.

Note: For information on the return codes that give you feedback on the **RSFlowInvoke** or **JRSFlowInvoke** runs, see "RSFlowInvoke and JRSFlowInvoke results" on page 73.

RSFlowInvoke and **JRSFlowInvoke** are available in the OO SDK home directory. They are also available in the OO home directory, in the **Studio\tools** folder. You can run **RSFlowInvoke** from any location, but you have to launch **JRSFlowInvoke** from the previously mentioned folder or from a path that includes that folder, as it is depending on several java libraries.

The basic **RSFlowInvoke.exe** syntax is:

```
RSFlowInvoke.exe {-host <hostname>:<port number> -flow <flow name>
| <URL>} [-inputs <input name>=<value>] [-u <user>]
-p <password>|-ep <encrypted password>] [-async]
[-rc <number of retries>] [-rw <number of seconds>]
[-t <timeout>] [-v]
```

The basic **JRSFlowInvoke.jar** syntax is:

```
java -jar JRSFlowInvoke.jar {-host <hostname>:<port>
-flow <flow name>| <URL>}
[-inputs <input name>=<value>] [-u <user>]
[-p <password>|-ep <encrypted password>] [-async]
[-rc <number of retries>] [-rw <number of seconds>]
```

You can reference the flow in **RSFlowInvoke** or **JRSFlowInvoke** using:

- The `-host` and `-flow` options.
- A URL. The URL must have the correct format for managing a flow and be enclosed in quotation marks. For information on building a correctly-formatted URL, see "Creating a URL for running a flow" on page 61.

Option Syntax

The **RSFlowInvoke** and **JRSFlowInvoke** options are:

- `-async`
  Specifies that the flow runs asynchronously, returning a result before completing its run.
- `-ep <encrypted password>`
  Specifies the encrypted password for the host. Do not use this option if you use the `-p` option to specify an unencrypted password. For information about creating an encrypted password from

within **RSFlowInvoke**, see "Creating an encrypted password" on page 71.

- `-flow <flow name>`
  Specifies the flow name or UUID.
- `-host <hostname>:<port number>`
  Specifies the host name and port number, separated by a colon.
- `-inputs <input name>=<value>`
  Specifies the inputs for the flow, using the `name=value&name2=value2` format .If any of the inputs or their values contain a space, wrap the `name=value&name2=value2` in quotes: `"name=value&name2=values"`.

  Note: To use it with **RSFlowInvoke** or **JRSFlowInvoke**, an input must have a flow variable assigned to it in the **Assign to Variable** drop-down box on the Studio **Input Summary**. This option is used when you specify host and flow options.
- `-p <password>`
  Specifies the password for the host. Do not use this option if you use the `-ep` option to specify an encrypted password.
- `-rc <number of retries>`
  Specifies the number of times to retry a flow that fails. The default value is `0`. The maximum value is `30`.
- `-rw <number of seconds>`
  Specifies the number of seconds to wait between retries. The default value is `5` seconds.
- `-t <timeout>`
  Specifies the timeout, in seconds. The default value is `100` seconds.
- `-u <username>`
  Specifies the host's user name.

- `-v` and `-verbose`
  Specifies that the entire output is to be written to the screen.

Using RSFlowInvoke or JRSFlowInvoke from a command line

The following examples demonstrate how to list and run flows using **RSFlowInvoke** or **JRSFlowInvoke** from a command line or from any application taking input from a command line.

- This example lists the flows specified in the URL:

```
java –jar JRSFlowInvoke.jar “https://localhost:8443/PAS/services/rest/list/MyFolder/”
-u rsadmin -ep BKmIQF6o0dItQkcUYNEeGw==
```

- This example runs the flow with input names and values specified in the URL:

```
java –jar JRSFlowInvoke.jar “https://localhost:8443/PAS/services/rest/run/Library/MyFolder/
TestFlow?inputName1=inputValue1&inputName2=inputValue2”
-u admin -ep BKmIQF6o0dItQkcUYNEeGw==
```

- This example lists the flows specified using the `–host` and `–flow` options:

```
java –jar JRSFlowInvoke.jar –host localhost:8443
–flow /PAS/services/rest/list/MyFolder/
-u admin -ep BKmIQF6o0dItQkcUYNEeGw==
```

- This example runs the flow with input names and values specified using the `–host` and `–flow` options:

```
java –jar JRSFlowInvoke.jar –host localhost:8443
–flow /PAS/services/rest/run/Library/MyFolder/TestFlow
-u rsadmin -ep BKmIQF6o0dItQkcUYNEeGw==
-inputs “inputName1=inputValue1&inputName2=inputValue2”
```

Using RSFlowInvoke or JRSFlowInvoke in a script or batch file

In a script or batch file, the syntax for using **RSFlowInvoke.exe** is the same as it is for using it in a command line. The syntax for **JRSFlowInvoke.jar** is slightly different.

- The **RSFlowInvoke.exe** syntax in a script or batch file is:

```
RSFlowInvoke.exe {–host <hostname>:<port number>
–flow <flow name>|<URL>} [–inputs <input name>=<value>]
[–u <user>] –p <password>|–ep <encrypted password>]
[–rc <number of retries>] [–rw <number of seconds>]
[–t <timeout>] [–v]
```

- The **JRSFlowInvoke.jar** syntax in a script or batch file is:

```
java –jar JRSFlowInvoke.jar {–host <hostname>:<port>
–flow <flow name>|<URL>} [–inputs <input name>=<value>]
[–u <user>] [–p <password>|–ep <encrypted password>]
[–rc <number of retries>] [–rw <number of seconds>]
```

To find the usage of the tool, type `java –jar JRSFlowInvoke.jar`.

Searching for a flow using JRSFlowInvoke.jar

You can use **JRSFlowInvoke.jar** to search for a flow outside Central. The search uses the **Apache Lucene** search syntax. For more information on this syntax, see the *Apache Software Foundation Query Parser Syntax Web page*.

Use the following syntax, which includes a properly formatted query string:

```
java -jar JRSFlowInvoke.jar "https://{<host>:<port>}/PAS/services/http/search? queryString = {<sequence_of_term_expressions>}" [-u <user>] [-p <password>]
```

The options are:

- `<host>` specifies the Central server where to perform the search.
- `<port>` specifies the port number on the Central server used for the Central - client communication.
- `<sequence_of_term_expressions>` is a search string, which can be:
  - A single term expression of the form: `<fieldname>:<term>`
  - A sequence of terms of the form:

    `<fieldname>:<term> + (<operator> + <sequence_of_term_expressions>)`

    - `<fieldname>` specifies one of the fields shown in the following table. These are not case sensitive.

Field	Description
`Category`	The category assigned to the flow.
`Description`	The flow description.
`Domain`	A domain term associated with the flow.
`ID`	The flow UUID.
`Input`	An input to an operation used in the flow.
`Name`	The flow name.

Field	Description
`Type`	The type of an operation used in the flow. The terms you can match in this field and the operation types they represent include: ◦ `cmd` - Command. ◦ `flow` - An operation which is a flow. ◦ `http` - HTTP, also known as shell. ◦ `other` - Scriptlet. ◦ `script.perl` - Perl script. ◦ `ssh` - SSH (Secure Shell). ◦ `telnet` - Telnet. ◦ `lock` - Acquire Lock. ◦ `unlock` - Release Lock.
`Stepdescription`	The description of one of the flow steps.
`Stepname`	The name of one of the flow steps.

  - `<term>` specifies the particular value of the field that can find the desired flow.
  - `<operator>` is one of the operators supported in the Apache Lucene search syntax: `AND, +, OR, NOT,` and `–`
- `-u <user>` specifies a user account which has the permissions to view and start a flow.
- `-p <password>` specifies the password for the user account.
- `-ep <encrypted password>` specifies the encrypted password for the user account. See "Creating an encrypted password" below for more information.

Example

This example searches for flows for which the field `Name` has a value of `John's Flow`.

```
java -jar JRSFlowInvoke.jar "https://{localhost:8443}/PAS/services/http/search? queryString = Name:John's Flow" -u admin -ep BKmIQF6o0dItQkcUYNEeGw==
```

To use the example, delete the text strings in the sample that don't fit your particulars and replace them with appropriate values.

Creating an encrypted password

RSFlowInvoke and **JRSFlowInvoke** allow you to send encrypted passwords over the Internet.

To create an encrypted password for use with **RSFlowInvoke** or **JRSFlowInvoke**, proceed to the following:

1. In a command window, type and run one of the following commands:

   ```
   RSFlowInvoke.exe –cp
   ```

or

```
java -jar JRSFlowInvoke.jar -cp
```

2. At the **Enter password** prompt, type the password.
3. At the **Enter password again** prompt, retype the password.

RSFlowInvoke and **JRSFlowInvoke** encrypt the password. When you run **RSFlowInvoke** or **JRSFlowInvoke** with the encrypted password, use the `-ep` option instead of the usual `-p` option for the password.

Registering RSFlowInvoke with the Global Assembly Cache

The **Global Assembly Cache** (GAC) is a store on a local .NET machine for .NET code assemblies. If you register **RSFlowInvoke.exe** with GAC, you can start a flow from within a .NET application, using any .NET compatible language (such as C#).

To register or unregister **RSFlowInvoke** in GAC:

1. On a .NET machine, open a command-line window and type in:

   ```
   gacutil.exe [/i|/u] RSFlowInvoke.exe
   ```

   Option syntax

   - `/i` registers **RSFlowInvoke.exe** with GAC.
   - `/u` unregisters **RSFlowInvoke.exe** with GAC.
2. Once **RSFlowInvoke.exe** is registered with GAC, type the following to view the assembly (compiled code) information:

   ```
   RSFlowInvoke.exe -s
   ```

The following is an example of the `RSFlowInvoke.exe -s` command output:

```
Assembly Name:
  RSFlowInvoke, Version=1.0.3098.16154, Culture=neutral, PublicKeyToken=4c09181d83b84dbc


Fully Qualified Type Name:
  RepairSystem.RSFlowInvoke


Method Name:
  ExecuteHeadlessFlow(
      System.String url,
      System.String username,
      System.String password,
      System.String authType,
      System.Boolean encryptedPassword)
```

RSFlowInvoke and JRSFlowInvoke results

The **RSFlowInvoke** and **JRSFlowInvoke** run information is specified through the return codes shown in the following table:

Return Code	Description
0	The flow was run. This code is not related to the flow response.
1	Central responded with HTTP code `503`. This usually means that Central lacked the resources needed to run the flow.
2	An unknown internal server error occurred in Central.
3	**RSFlowInvoke** was unable to authenticate against Central.
4	The specified URL or flow was not found.
5	A socket timeout occurred. A socket is a software object that connects an application to a network protocol.
6	An unknown socket (communication) error occurred.
7	An unknown error occurred.

Finding and running flows using the WSCentralService SOAP API

The **WSCentralService** service assures search and execution. The **WSCentralService** service provides a **SOAP API** that enables you to:

- Search for a flow by querying the flow library, using the criteria provided by a query string. Query strings are based on an **Apache Lucene** indexed search. For more information on the **Apache Lucene** search, see the *Apache Lucene Web page*.
- Control flow execution. This includes running, pausing, resuming, canceling a flow, and viewing the status of a flow run.

The **WSCentralService SOAP API** Java and .NET classes and interfaces are located in the OO SDK home directory, in the **lib** folder. The certificates, **keystore**, **WSDL**, and code samples are located in the OO SDK home directory, in the **samples** folder.

Accessing the WSCentralService WSDL

The **WSCentralService WSDL** describes the **WSCentralService** service and the operations it can perform. You can access the **WSCentralService WSDL** at:

```
https://central_server:8443/PAS/services/WSCentralService?wsdl
```

The `central_server` is the name of the server running Central. The samples included in the OO SDK home directory, in the **samples\client** folder, demonstrate how to use the service with Java and .NET.

WS Central Service: Using the API documentation

The **WSCentralService SOAP API** reference documentation is included in the OO SDK home directory, in the **docs\javadocs\WSCentralService** folder. The reference documentation covers the **WSCentralService** API, objects, and the `WSCentralServiceSession` wrapper class implementing the service public interface.

How WSCentralService manages security and authentication

WSCentralService supports HTTP basic authentication. The client must provide the user name and password of a user account that can run and manage flows that are started outside Central. The message context associated with a client session must include the user name and password. Otherwise, the session fails and the client is issued a security violation. For example, see the test client code **Util.java** in the OO SDK home directory, in the **samples\client\java\src** folder.

The service also supports **Kerberos** single sign-on authentication based on the `Oasis Web Services Security Kerberos Token Profile`. The format of the `BinarySecurityToken` node in the header security node is:

```
<soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
  <wsse:BinarySecurityToken

   EncodingType="wsse:Base64Binary"

   ValueType="wsse:Kerberosv5_AP_REQ"

   Id="CentralKrbSSOToken">YIIJAQYJKoZIhvcSAQICAQBuggjwMIII....

  </wsse:BinarySecurityToken>

 </wsse:Security>

</soapenv:Header>
```

The `Id` attribute of the `<wsse:BinarySecurityToken>` must be included and must be set to `CentralKrbSSOToken`. The name and value of the `Id` attribute are case sensitive.

WS Central Service: Importing the SSL Certificate

JAVA

In order to enable the service handling the SSL (handshake which begins a SSL session), you can:

- Import, into any **keystore** you choose, the **central.crt** security certificate included in the OO SDK home directory, in the **samples\client\java\resources** folder.

- Use the sample **keystore** provided in the **samples\client\java\resources\cacerts.sample**. A **keystore** is a file containing keys, certificates, and trusted roots. The root certificates of signing authorities are kept in a file called **cacerts**.

To import the certificate into the default **keystore** provided with the Java Runtime Environment (JRE), you need to:

- open a command window
- change the directory to the **lib\security** folder in the Java home directory
- run the following command under **$(java.home)\lib\security**:
  `keytool -import -alias pas -file central.crt -keystore cacerts`

When prompted for the JRE **cacerts** password, type: `changeit`

.NET

In order to setup SSL for running the .NET samples, the Central certificate must be placed in the **Personal** store of the client machine:

1. Take the public Central key from the **rc_keystore** keystore file corresponding to the `pas` alias and place it into a `DER` encoded file:
   `keytool -exportcert -alias pas -keystore rc_keystore -file pas_crt.der`
2. Install the certificate into the **Personal** store by selecting **Install Certificate** on the **pas_crt.der** file.
3. Use the **certmgr.msc** tool and check that the certificate was installed in the **Personal** store.

WS Central Service: Sample client code

You can find sample client code for **WSCentralService** in the OO SDK home directory, in the **samples\client\.Net\hp** and **samples\client\java\src** folders. The **lib\WSCentralService.jar** file contains a wrapper for the **WSCentralService** service, named `WSCentralServiceSession`. You can use this class in developing your client. It wraps the service API stubs and includes additional functionality for Java and .NET.

Running a JAVA sample from ANT

In order to run a sample, you need to go through the following steps:

- From the command line, inside the **samples\client\java** folder, run the `>ant` command for building the samples .
- From the command line, inside the **samples\client\java** folder, run the `>ant run.sample.usage` command to print the usage.
- From the command line, inside the **samples\client\java** folder, run the `>ant run.sample <arguments>` command.

For example, in order to run the **MoveFolder** sample using the following arguments:

- `source`: **Library/My Ops Flows/source/moveme**.
- `dest`:**Library/My Ops Flows/dest**.
- `overwrite:true`.

use the following command:

```
 >ant run.sample -Dsample.mainclass=MoveFolder -
Dsample.mainargs="'/Library/My Ops Flows/source/moveme' '/Library/My
Ops Flows/dest' 'true'"
```

WSCentralService: Service stubs sample

The following **sample.cmd** file generates Java service stubs. You can use these as an alternative to the supplied **WSCentralService.jar** file.

```
@echo off

Rem replace %axis-1_4% and %wsdl_path% with the actual paths on your machine.
Rem your path statement must include the folder where java.exe exist.



set JLIBS=%axis-1_4%\lib

set SERVICE_ADDRESS=%wsdl_path%\WSCentralService.wsdl

set SERVICE_PACKAGE=com.iconclude.dharma.services.wscentralservice.client



set BUILD_PATH=.

set CLASS_PATH=.

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\axis.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\jaxrpc.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\commons-codec.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\commons-httpclient.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\commons-logging-1.0.4.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\commons-discovery.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\wsdl4j-1.5.1.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\activation.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\saaj.jar

set CLASS_PATH=%CLASS_PATH%;%JLIBS%\mail.jar

set command1=java -classpath %CLASS_PATH% org.apache.axis.
wsdl.WSDL2Java -a -p %SERVICE_PACKAGE% -v
-o %BUILD_PATH% %SERVICE_ADDRESS%



%command1%
```

Resuming runs from the command line

When a flow run has been handed off, you can resume it from a command line. You can resume the run so that it completes either synchronously or asynchronously.

- Resuming a run synchronously means that after resuming the run, the Central service waits for it to complete before starting other runs.

  For information on resuming a run synchronously, see "Resuming a run synchronously" on page 79.

- Resuming a run asynchronously means that after resuming the run, the Central service does not wait for it to complete before starting other runs.

  For information on resuming a run asynchronously, see "Resuming a run asynchronously" on page 80.

Notes:

- Although you can resume a run that has not been paused, it is strongly recommended that you do not.
- When you resume a run, you identify it by its `run ID`, rather than by its `run history ID`. If you don't know the run ID, you can obtain it from OO Central on the **Current Runs** tab.

Suppose you have a flow that contains a couple of transitions marked for hand off:

Flow with transitions marked for handoff

The following example shows a command that starts a run of such a flow.

```
$ java -jar JRSFlowInvoke.jar
http://localhost:8080/PAS/services/http/run/Library/test/handoff/
flow-that-hands-off -u myuser -p *****
```

This action returns the following XML, in the block as presented below:

```
<?xml version="1.0" encoding="UTF-8"?>
<runResponse><runReturn>< item><name>runId </name><value>718032</value></item>
<item><name>runHistoryId</name>
<value>718031</value></item>
<item><name>runReportUrl</name>
<value>http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&amp;sp=SINDIVIDUAL_REPAIR_LEVEL&amp;
sp=S368a860a-a54a-4901-83d7-
11193ce2ca64&amp;sp=l0&amp;sp=l718031</value></item>
<item><name>displayRunReportUrl</name><value>
<![CDATA[http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&sp=SINDIVIDUAL_REPAIR_LEVEL&sp=S368a860a-a54a-
4901-83d7-11193ce2ca64&
```

```
sp=l0&sp=l718031]]></value></item>
<item><name>runStartTime</name><value>03/01/10 10:11</value></item>
<item><name>runEndTime</name><value>03/01/10 10:11</value></item>
<item><name>flowResponse</name><value>HANDOFF</value></item>
<item><name>flowResult</name><value>{}</value></item>
<item><name>flowReturnCode</name><value>Not a Return</value></item></runReturn></runResponse>
```

To clarify its structure and improve its readability, we'll format the above XML conventionally.

```
<?xml version="1.0" encoding="UTF-8"?>
<runResponse>
 <runReturn>
  <item>
   <name>runId</name>
   <value>718032</value>
  </item>
  <item>
   <name>runHistoryId</name>
   <value>718031</value>
  </item>
  <item>
   <name>runReportUrl</name>
   <value>http://localhost:8080/PAS/app?service=RCLinkService/
       ReportLinkDispatch&amp;sp=SINDIVIDUAL_REPAIR_LEVEL&amp;
     sp=S368a860a-a54a-4901-83d7-11193ce2ca64&amp;
       sp=l0&amp;sp=l718031</value>
  </item>
  <item>
   <name>displayRunReportUrl</name>
   <value><![CDATA[http://localhost:8080/PAS/app?service=RCLinkService/
      ReportLinkDispatch&sp=SINDIVIDUAL_REPAIR_LEVEL&
      sp=S368a860a-a54a-4901-83d7-11193ce2ca64&sp=l0&
       sp=l718031]]></value>
  </item>
  <item>
   <name>runStartTime</name>
   <value>03/01/10 10:11</value>
  </item>
  <item>
   <name>runEndTime</name>
   <value>03/01/10 10:11</value>
  </item>
  <item>
   <name>flowResponse</name>
   <value>HANDOFF</value>
  </item>
  <item>
   <name>flowResult</name>
   <value>{}</value>
  </item>
  </item>
  <item>
   <name>flowReturnCode</name>
   <value>Not a Return</value>
  </item>
 </runReturn>
</runResponse>
```

Observations about the preceding XML:

- The run ID is the value for the first `item` element within the `runReturn` element. The run ID is in this case `718032`.

- The `flowResponse` item shows us that the run was stopped in a `HANDOFF` state because a handoff transition was followed.
- The `flowReturnCode` item has the value `Not a Return` because no return step was ever reached and thus the flow run is not complete.

Resuming a run synchronously

The following command resumes the run headlessly in synchronous mode:

```
$ java -jar JRSFlowInvoke.jar http://localhost:8080/PAS/services/http/resume/718032
 -u myuser -p *****
```

Note: `resume` is the verb which determines that the flow is resumed in synchronous mode.

The following XML is returned:

```
<resumeResponse><resumeReturn><item><name>runId
</name><value>718032</value></item>
<item><name>runHistoryId</name>
<value>718031</value></item>
<item><name>runReportUrl</name>
<value>http://localhost:8080/PAS/
app?service=RCLinkService/
ReportLinkDispatch&amp;sp=SINDIVIDUAL_REPAIR_LEVEL&&
sp=S368a860a-a54a-4901-83d7-11193ce2ca64&&
sp=l0&&sp=l718031</value></item>
<item><name>displayRunReportUrl</name><value>
<![CDATA[http://localhost:8080/PAS/app?service=
RCLinkService/ReportLinkDispatch&sp=SINDIVIDUAL_REPAIR_LEVEL&
sp=S368a860a-a54a-4901-83d7-11193ce2ca64&
sp=l0&sp=l718031]]></value></item>
<item><name>runStartTime
</name><value>03/01/10 10:11</value>
</item><item><name>runEndTime</name>
<value>03/01/10 10:12</value></item>
<item><name>flowResponse</name>
<value>HANDOFF</value></item>
<item><name>flowResult</name>
<value>{}</value></item>
<item><name>flowReturnCode</name><value>Not a
Return</value></item></resumeReturn></resumeResponse>
```

The returned XML is very similar to the one from the run initiation. The `flowResponse` item indicates that the run was stopped again in a `HANDOFF` state, due to the second handoff transition being followed.

Resume the run once again, using:

```
$ java -jar JRSFlowInvoke.jar http://localhost:8080/PAS/services/http/resume/718032 -u myuser
-p *****
```

The following XML is returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<resumeResponse><resumeReturn>
<item><name>runId</name><value>718032</value>
</item>
<item><name>runHistoryId</name>
<value>718031</value></item>
```

```
<item><name>runReportUrl</name>
<value>http://localhost:8080/PAS/
app?service=RCLinkService/
ReportLinkDispatch&amp;sp=SINDIVIDUAL_REPAIR_LEVEL&amp;sp=S368a860a-a54a-4901-83d7-
11193ce2ca64&amp;sp=l0&amp;
sp=l718031</value></item>
<item><name>displayRunReportUrl</name>
<value><![CDATA[http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&sp=SINDIVIDUAL_REPAIR_LEVEL&sp=S368a860a-a54a-4901-83d7-11193ce2ca64&sp=l0&

sp=l718031]]></value></item>
<item><name>runStartTime</name>
<value>03/01/10 10:11</value></item>
<item><name>runEndTime</name>
<value>03/01/10 10:14</value></item>
<item><name>flowResponse</name>
<value>success</value></item><item>
<name>flowResult</name><value>
{FailureMessage=;TimedOut=;Result=;}</value></item>
<item><name>flowReturnCode</name><value>Resolved</value></item></resumeReturn>
</resumeResponse>
```

This time the flow completed, the response is `success` and the return code is `Resolved`.

Resuming a run asynchronously

To resume a run asynchronously, the links are very similar to the synchronous mode. The only difference is that the `resume` verb is replaced with `resume_async`.

The initial run invocation is:

```
$ java -jar JRSFlowInvoke.jar
http://localhost:8080/PAS/services/http/run/Library/test/handoff/
flow-that-hands-off -u myuser -p *****
```

The run encounters the first handoff transition, and returns the following XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<runResponse><runReturn><item><name>runId</name>
<value>718067</value></item>
<item><name>runHistoryId</name>
<value>718066</value></item>
<item><name>runReportUrl</name><value>
http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&amp;
sp=SINDIVIDUAL_REPAIR_LEVEL&amp;sp=S368a860a-a54a-4901-83d7-11193ce2ca64&amp;
sp=l0&amp;sp=l718066</value></item><item>
<name>displayRunReportUrl</name>
<value><![CDATA[http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&
sp=SINDIVIDUAL_REPAIR_LEVEL&sp=S368a860a-a54a-4901-83d7-11193ce2ca64&sp=l0&sp=l718066]]>
</value></item>
<item><name>runStartTime</name><value>03/01/10 10:16</value></item>
<item><name>runEndTime</name><value>03/01/10 10:16</value></item>
<item><name>flowResponse</name><value>HANDOFF</value></item>
<item><name>flowResult</name><value>{}</value></item>
<item><name>flowReturnCode</name><value>Not a Return</value></item>
</runReturn></runResponse>
```

In this case, the run ID is `718067`.

Resume the run asynchronously, using the `resume_async` verb:

```
$ java -jar JRSFlowInvoke.jar
http://localhost:8080/PAS/services/http/resume_async/718067 -u myuser -p *****
```

The following XML is returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<resumeResponse><resumeReturn>
<item><name>runId</name><value>718067</value></item>
<item><name>runHistoryId</name><value>718066</value></item>
<item><name>runReportUrl</name>
<value>http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&amp;
sp=SINDIVIDUAL_REPAIR_LEVEL&amp;sp=S368a860a-a54a-4901-83d7-11193ce2ca64&amp;
sp=l0&amp;sp=l718066</value></item>
<item><name>displayRunReportUrl</name>
<value><![CDATA[http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&sp=SINDIVIDUAL_REPAIR_LEVEL&
sp=S368a860a-a54a-4901-83d7-11193ce2ca64&
sp=l0&sp=l718066]]></value></item>
<item><name>runStartTime</name><value>03/01/10 10:16</value></item>
<item><name>flowResponse</name><value></value></item>
<item><name>flowResult</name><value></value></item>
<item><name>flowReturnCode</name><value>Not a Return</value></item>
</resumeReturn></resumeResponse>
```

The response is very similar to the synchronous resumption of the run, with the following differences:

- There is no `runEndTime` item because the end time is unknown.
- The `flowResponse` item is empty.

Run the asynchronous resumption again:

```
$ java -jar JRSFlowInvoke.jar http://localhost:8080/PAS
/services/http/resume_async/718067
 -u myuser -p *****
```

The returned XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
<resumeResponse><resumeReturn>
<item><name>runId</name><value>718067</value></item>
<item><name>runHistoryId</name><value>718066</value></item>
<item><name>runReportUrl</name>
<value>http://localhost:8080/PAS/app?service
=RCLinkService/ReportLinkDispatch&amp;
sp=SINDIVIDUAL_REPAIR_LEVEL&amp;sp=
S368a860a-a54a-4901-83d7-11193ce2ca64&amp;sp=l0&amp;
sp=l718066</value></item>
<item><name>displayRunReportUrl</name><value>
<![CDATA[http://localhost:8080/PAS/app?service=RCLinkService/
ReportLinkDispatch&sp=SINDIVIDUAL_REPAIR_LEVEL&
sp=S368a860a-a54a-4901-83d7-11193ce2ca64&sp=l0&sp=l718066]]></value></item>
<item><name>runStartTime</name><value>03/01/10 10:16</value></item>
<item><name>flowResponse</name><value></value></item>
<item><name>flowResult</name><value></value></item>
<item><name>flowReturnCode</name><value>Not a Return</value></item>
</resumeReturn></resumeResponse>
```

Again, the answer is the same.

Resuming the flow for a third time, as the flow should have finished in the background, running it should produce an error:

```
$ java -jar JRSFlowInvoke.jar http://localhost:8080/PAS/services/http/resume_async/718067 -u
myuser -p *****
```

The error looks like the following:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Error 404 request=/PAS/services/http/
resume_async/718067, error=path not found: Run: 718067 not found, either finished or deleted.,
host=127.0.0.1</title>
</head>
<body><h2>HTTP ERROR: 404</h2><pre>request=/PAS/services/http/resume_async/718067,
error=path not found: Run: 718067 not found, either finished or deleted., host=127.0.0.1</pre>
<p>RequestURI=/PAS/services/http/resume_async/718067</p>
<p><i><small><a href="http://jetty.mortbay.org/">Powered by LaunchJetty6://</a>
</small></i></p>
<br/>
</body>
</html>
```

Note that resuming the flow in synchronous mode would obtain the same response:

```
$ java -jar JRSFlowInvoke.jar http://localhost:8080/PAS/services/http/resume/718067
-u myuser -p *****
```

The error is:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Error 404 request=/PAS/services/http/resume/718067,
error=path not found: Run: 718067 not found, either finished or deleted.,
host=127.0.0.1</title>
</head>
<body><h2>HTTP ERROR 404</h2><pre>request=/PAS/services/http/resume/718067,
error=path not found: Run: 718067 not found, either finished or deleted., host=127.0.0.1</pre>
<p>RequestURI=/PAS/services/http/resume/718067</p><p><i><small>
<a href="http://jetty.mortbay.org/">Powered by LaunchJetty6://</a></small></i></p>
<br/>
</body>
</html>
```

Chapter 4

Working with repositories from outside Studio

If you work with OO repositories frequently, you might find it easier and less time consuming to perform common OO repository functions outside Studio.

The OO SDK **Repository Tool** (**RepoTool**) is a thin wrapper that includes all the java dependencies and configuration files. It is a JAR file called **RepoTool.jar**. You need a **Java Runtime Environment** (JRE) to run it. You can use it to perform the following repository functions outside Studio:

- "Publishing a repository" on page 89. Publish new or changed OO objects, including flows and operations, from a source repository to a target repository.
- "Updating from a repository" on page 91. Update a source repository with new or changed OO objects from a target repository.
- "Publishing and updating a repository simultaneously" on page 91. Publish and update in one operation.
- "Exporting a repository" on page 92.
- "Verifying a repository" on page 92. Verify a repository, finding problems with the OO objects in it, and optionally fixing them.
- "Upgrading a repository" on page 93. Upgrade a repository to the latest version.
- "Encrypting a repository" on page 93, "Decrypting a repository" on page 94, and "Re-encrypting a repository" on page 94.
- "Setting default permissions for a repository" on page 95.
- "Importing a localization file " on page 96. Import a file with localized content.
- "Exporting content to be localized" on page 95. Export the content to be localized, to a file .
- "Setting flags" on page 96. Set the sealed, hidden, and content flags for the objects in a specified path.
- "Deleting objects" on page 97. Delete objects in a specified path.

Note: The target repository is always a Central public repository.

Using the Repository Tool

Run the **RepoTool** from a command line, using one of the 10 primary options. Each **RepoTool** primary option indicates the repository function to perform. **RepoTool** also has secondary options, providing information required by the primary options.

- For information on the primary options, see "Primary RepoTool options" below.
- For information on the secondary options, see "Secondary RepoTool options" on next page.

Syntax: `java jar RepoTool.jar [<primary option>] [<secondary options>]`

The **RepoTool.jar** file is located in the OO SDK home directory, where it is entirely self-contained.

To see the list of parameters, run it with no arguments: `java -jar RepoTool.jar`

Primary RepoTool options

The primary **RepoTool** options specify the repository functions to be performed. In the following table, click an option name to see more information on the action it launches.

Primary Option	Description
-publish	Publishes changes from a source repository to a target repository.
-update	Updates from a target repository to a source repository.
-publishupdate	A combination of `-publish` and `-update`.
-export	Exports a source repository, making flows and OO objects available to users who do not share the same public repository as you.
-verify	Verifies the structural integrity of a repository, then lists and optionally fixes any problems. The repository must be a local repository.
-upgrade	Upgrades a repository to the latest version.
-encrypt	Encrypts a repository using an encryption password and saves it to another repository. The encrypted repository must be local.
-decrypt	Decrypts an encrypted repository using an encryption password and saves it to another repository. The decrypted repository must be local.
-reencrypt	Re-encrypts a local repository using an encryption password and encrypts it to another repository using another encryption password. The re-encrypted repository must be local.
-defaultperms	Sets default permissions for a repository. The repository must be local.
-localizeimport	Reads a localization file and places the strings into the content. The repository must be local.
-localizeexport	Exports localizable content to a file.
-setflags	Sets flags based on the settings of the `-seal`, `-hide`, and `-content` secondary options. The repository must be local.
-delete	Deletes objects based on the specified `-includepath`. The repository must be local.

Secondary RepoTool options

Most of the secondary **RepoTool** options work with more than one primary option. If a secondary option works with only one primary option, it is described in the section that explains how to use the corresponding primary option.

The rest of the secondary **RepoTool** options and the primary options they work with are shown in the following table:

Secondary Option	Description	Used With

`-1 <repository1>`	Specifies the `<repository1>` path and name. For most of the primary **RepoTool** options, this is the source repository.
	For the following options, `<repository1>` can only be a local repository:
	`-verify`
	`-upgrade`
	`-encrypt`
	`-decrypt`
	`-reencrypt`
	`-delete`
	`-localizeimport`
	`-localizeexport`
	For the following options, `<repository1>` can be a local or Central repository:
	`-publish`
	`-update`
	`-publishupdate`
	`-export`
	`-defaultperms`
	A local repository is specified by a path. For example, **C:\MyFolder\MyRepository**
	A Central repository is specified by a URL. For example, **https://central-host2:8443**
		`-publish`
		`-update`
		`-publishupdate`
		`-export`
		`-verify`
		`-upgrade`
		`-encrypt`
		`-decrypt`
		`-reencrypt`
		`-defaultperms`
		`-localizeexport`
		`-localizeimport`
		`-setflags`
		`-delete`

<table>
<tr><th>Secondary Option</th><th>Description</th><th>Used With</th></tr>
<tr><td>-2 <repository2></td><td>Specifies the <repository2> path and name. For most of the primary RepoTool options, this is the target repository.

For the following options, <repository2> can only be a local repository:

-encrypt

-decrypt

-reencrypt

For the following options, <repository2> can be a local or Central repository:

-publishupdate

-export

A local repository is specified by a path. For example, C:\MyFolder\MyRepository

A Central repository is specified by a URL. For example, https://central-host2:8443</td><td>-publish

-update

-publishupdate

-export

-encrypt

-decrypt

-reencrypt</td></tr>
<tr><td>-c <value></td><td>Specifies how conflicts are resolved. A conflict can occur if changes have been made to the same object in both the target and source repositories.
The <value> is one of the following:

<table><tr><th>Value</th><th>Effect</th></tr><tr><td>0</td><td>Skips the conflicts.</td></tr><tr><td>r1</td><td>Conforms <repository1> to <repository2>.</td></tr><tr><td>r2</td><td>Conforms <repository2> to <repository1>.</td></tr></table>
See "Publishing a repository" on page 89 for more information on using the -c option.</td><td>-publish

-update

-publishupdate</td></tr>
<tr><td>-children</td><td>When used with the -delete secondary option,
-children deletes only the children of the -includepath specified paths.</td><td>-delete</td></tr>
</table>

Secondary Option	Description	Used With
`-content <true|false>`	All objects specified by `-includepath` will have their content flag set to the selected value `<true|false>`. Folders are recursive.	`-setflags`

`-excludepath <path>`	Specifies a `<repository1>` path to exclude when publishing or updating. For example, `-excludepath "/Library/My Repairs"`. The path must be enclosed in quotation marks. The flows and objects in the excluded path will not be published or updated.	`-publish`
		`-update`
		`-publishupdate`

`-hide <true|false>`	All folders and flows specified by `-includepath` will have their hidden flag set to this value. If set to `true`, the objects are hidden in Central. Folders are not recursive.	`-setflags`

`-includepath <path>`	Specifies the only path in `<repository1>` to include when publishing or updating. For example, `-includepath "/Library/My Repairs"`. Only the path specified in `<repository1>` will be published or updated.	`-publish`
		`-update`
		`-publishupdate`

`-includereferences`	You can only use this option when you use the `-includepath`. The `-includereferences` option specifies that any flow or operation used by the flows in the `-includepath` option will also be published.	`-publish`
		`-update`
		`-publishupdate`

`-localizationfile <localizationfile>`	Specifies the file from which to read or the file to write to for localization.	`-localizeimport` `-localizeexport`

`-loginurl <loginurl>`	Specifies the URL of the Central server with which to authenticate if `<repository1>` or `<repository2>` is remote.	`-publish`
		`-update`
		`-publishupdate`
		`-export`
		`-localizeexport`
		`-localizeimport`

Secondary Option	Description	Used With

`-n`	Specifies that **RepoTool** should not perform the `-publish`, `-update`, `-publishupdate`, or `-export` option with which the `-n` option is used. Instead, print the results that would occur if the option was performed.	`-publish`
		`-update`
		`-publishupdate`
		`-export`

`-p <password>`	Specifies the password for the Central or remote server.	`-publish`
		`-update`
		`-publishupdate`
		`-export`
		`-upgrade`
		`-encrypt`
		`-decrypt`
		`-defaultperms`
		`-localizeexport`
		`-localizeimport`

`-q <repo2password>`	Specifies the encryption password for `<repository2>`. It is ignored if `<repository2>` is a remote repository or if it is not encrypted.	`-publish`
		`-encrypt`
		`-decrypt`
		`-reencrypt`

`-r <repo1password>`	Specifies the encryption password for `<repository1>`. It is ignored if `<repository1>` is a remote repository or if it is not encrypted.	`-publish`
		`-update`
		`-publishupdate`
		`-export`
		`-verify`
		`-upgrade`
		`-decrypt`
		`-reencrypt`
		`-defaultperms`
		`-localizeexport`
		`-localizeimport`

Secondary Option	Description	Used With
`-seal <true|false>`	All folders specified by `-includepath` will have their sealed flag set to the selected value `<true|false>`. If set to `true`, the folders are sealed. When a folder is sealed, it cannot be changed. Folders are recursive. The repository must be local.	`-setflags`

`-u <username>`	Specifies the username for the Central or remote server.
	
	Note: The user name must belong to an administrator for the options shown in the **Used With** column. An exception is the `-publish` option, where the user name can belong to a member of the `PROMOTER` group.
		`-publish`
		`-update`
		`-publishupdate`
		`-export`
		`-upgrade`
		`-encrypt`
		`-decrypt`
		`-defaultperms`
		`-localizeexport`
		`-localizeimport`

Return codes

The following table contains **RepoTool** return codes and their meaning.

Return code	Meaning
0	Success.
1	An exception happened during publishing.
2	The repository changed while publishing.
3	Bad command line options were given.
4	There is no difference between the source and target repository.

Publishing a repository

The `-publish` option copies new or changed objects, such as flows and operations, from a source repository (`<repository1>`) to a target repository (`<repository2>`).

Syntax

```
java -jar RepoToo.jar -publish -loginurl <loginurl> -u <username> -p <password>
-1 <repository1> [-r <repo1password>] -2 <repository2>
[-q <repo2password>] -c 0|r1|r2 [-n] [-excludepath <path>] [-includepath <path>]
[-includereferences]
```

Both `<repository1>` and `<repository2>` stand for local repository paths or Central repository URLs.

If a conflict is reported due to changes to an object having the same name in the target and source repositories, **RepoTool** acts based on the `-c` option.

`-c` option	Effect
0	Skip the conflicts.
r1	Conform `<repository1>` to `<repository2>`.
r2	Conform `<repository2>` to `<repository1>`.

The following scenario illustrates how this works. Suppose that:

1. The `<repository1>` local repository and the `<repository2>` Central repository are synchronized and contain a flow named **testflow**.
2. You import a new version of **testflow** to `<repository1>`.
3. From a second local repository, you publish another version of **testflow** to `<repository2>`. Now **testflow** has changed in both `<repository1>` and `<repository2>`. A conflict will be reported for **testflow** when you preview publishing from `<repository1>` to `<repository2>`.
4. Use the `-c <value>` option to specify how you want to resolve a potential conflict.

The values for the `-c` option, and descriptions of the other secondary options used in the `-publish` syntax, are shown in "Primary RepoTool options" on page 84.

For additional information on how to publish a repository using Studio, see the *Studio Authoring Guide* or the online *Studio Help System*.

The following examples show some of the ways you can use the **RepoTool** `-publish` option.

Example 1

This example publishes the contents of the **C:\MyFolder\export** exported repository to the Central server **central-host2**. **RepoTool** acts on the `-c r2` option. If conflicts occur, it changes **central-host2** to resolve them.

```
java -jar RepoTool.jar -publish -loginurl https://central-host2:8443 -u admin
-p iconclude -1 c:\MyFolder\export -2 https://central-host2:8443 -c r2
```

Note: This is comparable to connecting Studio to **central-host2** and to the repository at **C:\MyFolder\export**.

Example 2

This example publishes the contents of the **Library/My Ops Flows/Network Flows** folder from the Central host **central-host1** to the Central host **central-host2**.

```
java -jar RepoTool.jar -publish -loginurl https://central-host1:8443 -u admin -p iconclude -1
https://central-host1:8443 -2 https://central-host2:8443 -c r2 -includepath "/Library/My Ops
Flows/Network Flows"
```

Updating from a repository

The `-update` option is the opposite of the `-publish` option. When you update a source repository (`<repository2>`) to a target repository (`<repository1>`), all the flows and objects that are new or have changed in the source repository are copied to the target repository.

Syntax

```
java -jar RepoTool.jar  -update -loginurl <loginurl> -u <username>
-p <password> -1 <repository1> [-r <repo1password>]
-2 <repository2> -c 0|r1|r2 [-n]
[-excludepath <path>] [-includepath <path>] [-includereferences]
```

The `<repository1>` parameter represents the repository from which you initiate the update. Both `<repository1>` and `<repository2>` can be local repository paths or Central repository URLs.

The secondary options used in the `-update` syntax are described in "Secondary RepoTool options" on page 85.

For additional information on how to to update a repository using Studio, see the *Studio Authoring Guide*.

Example

In this example, **RepoTool** updates the **Library/My Ops Flows/TestFlow** path in the **central2** repository from the **My Repository** local repository . The `-includereferences` option tells **RepoTool** to include all the references to the flows and operations used by **TestFlow**. The `-c` option tells **RepoTool** to change the **central2** repository to resolve any occurring conflicts.

```
java -jar RepoTool.jar  -update -1 https://central2:8443
-2 c:\MyFolder\My Repository -includepath "/Library/My Ops Flows/TestFlow" -includereferences
-cr1
```

Publishing and updating a repository simultaneously

The `-publishupdate` option copies objects that are new or have changed, from a source repository (`<repository1>`) to a target repository (`<repository2>`). It also copies flows and objects that are new or have changed, from a target repository (`<repository1>`) to a source repository (`<repository2>`).

Syntax

```
java -jar RepoTool.jar  -publishupdate -loginurl <loginurl>
-u <username> -p <password> -1 <repository1> [-r <repo1password>]
-2 <repository2> -c 0|r1|r2 [-n] [-excludepath <path>]
[-includepath <path>] [-includereferences]
```

The secondary options used in the `-publishupdate` syntax, are described in "Secondary RepoTool options" on page 85.

Exporting a repository

To make flows and OO objects available to authors with whom you do not share a public repository, you can use the `-export` option. This exports a repository (`<repository1>`) to a target location (`<repository2>`). The exported repository can be a local repository or a Central repository. The target location specifies a directory on the local file system. Other Studio authors can later import the repository from the target location.

Note: The **RepoTool** `-export` option creates `<repository2>`. The `<repository2>` directory should not exist prior to exporting `<repository1>`.

Syntax

```
java –jar RepoTool.jar  -export -loginurl <loginurl>
-u <username> -p <password> -1 <repository1>
[-r <repo1password>] -2 <repository2> [-x <path1>
-x <path2>] [-n]
```

The `-x <path>` option sets the path in the repository or library to be exported. For example, `-x "/Library/My Repairs/"`. The path specified in the `-x` option must be enclosed in quotation marks. You can specify multiple paths using more than one `-x` option. The default is `-x "/Library" -x "/Configuration/"`.

The other secondary options used in the `-export` syntax, are described in "Secondary RepoTool options" on page 85.

For additional information on how to export a repository using Studio, see the *Studio Authoring Guide* or the online *Studio Help System*.

Example

This example exports the **Library/My Ops Flows/Network Flows** and **Library/My Ops Flows/Database Flows** folders from the Central host repository **central-host1** to a new local repository, **My Repository**.

```
java –jar RepoTool.jar  -export -loginurl https://central-host1:8443 -u
admin -p iconclude -1 https://central-host1:8443 -2 c:\MyFolder\My Repository -x "/Library/My
Ops Flows/Network Flows/" -x "/Library/My Ops Flows/Database Flows/"
```

Note: This is comparable to connecting Studio to **central-host1**, selecting the **Library/My Ops Flows/Network Flows** and **Library/My Ops Flows/Database Flows** folders, and exporting them to **C:\MyFolder\My Repository**.

Verifying a repository

The `-verify` option allows you to verify the structural integrity of a local repository.

Syntax

```
java –jar RepoTool.jar -verify [-f] -1 <repository1>
[-r <repo1password>]
```

The `-f` option specifies that **RepoTool** will attempt to fix any structural integrity problems it encounters in `<repository1>`.

The other secondary options used in the `-verify` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example verifies the **C:\MyFolder\MyRepo** repository, lists any problems that exist in the flows and other objects in the repository, and then attempts to fix the problems.

```
java –jar RepoTool.jar -verify -f -1 C:\MyFolder\MyRepo
```

Upgrading a repository

The `-upgrade` option upgrades a local repository to the latest version. This option is designed to be used mainly by OO authors, as upgrades are normally done automatically in a production environment.

Syntax

```
java –jar RepoTool.jar -upgrade -u <username> -p <password> -1
<repository1> [-r <repo1password>]
```

The secondary options used in the `-upgrade` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example upgrades the **C:\MyFolder\MyRepo** repository to the latest **RepoTool.jar** associated version .

```
java –jar RepoTool.jar -upgrade -1 C:\MyFolder\MyRepo
```

Encrypting a repository

The `-encrypt` option makes a copy of a local repository `(<repository1>)`, encrypts it, and then saves it as `<repository2>`. You can use encryption to protect your repository from unauthorized users.

Syntax

```
java –jar RepoTool.jar -encrypt [-u <username> -p <password>] -1
<repository1> [-r <repo1password>] -2 <repository2> -q <repo2password>
```

If you modify, publish, update, import, or export `<repository2>`, you must use the `<repo2password>` password.

The secondary options used in the `-encrypt` syntax are described in "Secondary RepoTool options" on page 85.

For additional information on how to encrypt a repository using Studio, see the *Studio Authoring Guide* or the online *Studio Help System*.

Example

This example copies the **My Repository** repository , encrypts it, and then saves it as **My Encrypted Repository**.

```
java –jar RepoTool.jar -encrypt -1 c:\MyFolder\My Repository -r
iconclude -2 c:\MyFolder\My Encrypted Repository -q iconclude2
```

Decrypting a repository

The `–decrypt` option decrypts an encrypted repository (`<repository1>`) and saves it as a decrypted repository (`<repository2>`).

Syntax

```
java –jar RepoTool.jar -decrypt [-u <username> -p <password>] -1
<repository1> -r <repo1password> -2 <repository2>
```

For `<repo1password>`, use the password you set when you encrypted the repository. For additional information, see "Encrypting a repository" on previous page.

The secondary options used in the `–decrypt` syntax are described in "Secondary RepoTool options" on page 85.

For additional information on how to decrypt a repository using Studio, see the *Studio Authoring Guide* or the online *Studio Help System*.

Example

This example decrypts **My Encrypted Repository** using the `iconclude2` password and saves it as the decrypted repository **My Decrypted Repository**.

```
java –jar RepoTool.jar -decrypt -1 c:\MyFolder\My Encrypted Repository
-q iconclude2 -2 c:\MyFolder\My Decrypted Repository
```

Re-encrypting a repository

The `–reencrypt` option allows you to create a second encrypted copy of a repository with a different password.

Syntax

```
java –jar RepoTool.jar -reencrypt -1 <repository1> -r <repo1password>
-2 <repository2> -q <repo2password>
```

Using `–reencrypt`, **RepoTool** makes and encrypts a second copy of the encrypted repository `<repository1>`, named `<repository2>`. **RepoTool** also re-encrypts it with a new password, specified in `<repo2password>`.

The secondary options used in the `–reencrypt` syntax are described in "Secondary RepoTool options" on page 85.

For additional information on how to re-encrypt a repository using Studio, see the *Studio Authoring Guide* or the online *Studio Help System*.

Example

This example creates a second copy of **My Encrypted Repository**, named **My Second Encrypted Repository**, with the new password `iconclude4`.

```
java –jar RepoTool.jar -reencrypt -1 c:\MyFolder\My Encrypted
Repository –r iconclude2 -2 c:\MyFolder\My Second Encrypted Repository
-q iconclude4
```

Setting default permissions for a repository

The `–defaultperms` option allows you to set the default access permissions for `<repository1>`. This applies the default permissions to the entire `<repository1>` content. The default permissions for a newly created object, for the `EVERYBODY` group, are: `Read`, `Write`, `Execute`, and `Link To`.

Syntax

```
java –jar RepoTool.jar -defaultperms -u <username> -p <password> -1
<repository1> [-r <repo1password>]
```

For additional information on Studio access permissions, see the *Studio Authoring Guide* or the online *Studio Help System*.

The secondary options used in the `–defaultperms` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example sets the **My Repository** default permissions so that all the users in the `EVERYBODY` group have read, write, execute, and link permissions for it.

```
java –jar RepoTool.jar -defaultperms -1 "My Repository"
```

Exporting content to be localized

The `–localizeexport` option exports specified repository values to the file specified in the `–localizationfile` secondary output, so that it can be localized manually. This localized content can be later imported into the repository, using the `–localizeimport` option.

Syntax

```
java –jar RepoTool.jar -localizeexport [-loginurl <loginurl>]
[-u <username> -p <password>] -1 <repository1> [-r <repo1password>]
-localizationfile <localizationfile>
```

The secondary options used in the `–localizeexport` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example exports the localizable content in the **central-host1** repository to the **Localfile** localization file.

```
java –jar RepoTool.jar -localizeexport -loginurl https://central-
host1:8443 -u admin -p iconclude -1 https://central-host1:8443
-localizationfile Localfile
```

Importing a localization file

The `–localizeimport` option reads a localization file and places the localized strings in a Studio repository. The target Studio repository is specified by the `–1` secondary option. The localization file contains OO content that has been translated into another language. This option replaces the content from the specified repository with the localized content.

Syntax

```
java –jar RepoTool.jar  -localizeimport [-loginurl <loginurl>]
[-u <username> -p <password>] -1 <repository1> [-r <repo1password>]
-localizationfile <localizationfile>
```

The secondary options used in the `– localizeimport` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example reads the **Localfile** localization file and places the localized strings in the **central-host1** Studio repository.

```
java –jar RepoTool.jar -localizeimport
-loginurl https://central-host1:8443
-u admin -p iconclude -1 https://central-host1:8443
-localizationfile Localfile
```

Setting flags

The `–setflags` option sets flags for all the objects specified by the `–includepath` secondary option. This is based on the settings of the `–seal, –hide,` and `–content` secondary options. The repository must be local.

Syntax

```
java –jar RepoTool.jar -setflags -1 <repository1> -includepath <path>
[-includepath <path>] [-seal <true|false>] [-hide <true|false>]
[-content <true|false>]
```

The secondary options used in the `–setflags` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example sets the flags shown below for the **Library/My Ops Flows/Network Flows** repository.

- `-seal true` specifies that all the objects in the path have their `sealed` flag set to `true`.
- `-hide false` specifies that all the files and folders in the path have their `hidden` flag set to `false`.
- `-content true` specifies that all the objects in the path have their `content` flag set to `true`.

```
Java –jar RepoTool.jar -setflags -1 c:\MyFolder\My Repository
-includepath "/Library/My Ops Flows/Network Flows" -seal true -hide
false -content true
```

Deleting objects

The `–delete` option deletes the objects specified in the `–includepath` secondary option.

Syntax

```
java –jar RepoTool.jar -delete -1 <repository1> -includepath <path>
[-includepath <path>] [-children]
```

The secondary options used in the `–delete` syntax are described in "Secondary RepoTool options" on page 85.

Example

This example deletes only the children of the **Library/My Ops Flows/Network Flows** path.

```
java –jar RepoTool.jar -delete -1 c:\MyFolder\My Repository
-includepath "/Library/My Ops Flows/Network Flows" -children
```

Chapter 5

Packaging content

The **OO Content Packager** allows you to package content (repositories and RAS libraries) into **content modules**, and then install the packaged content on Central and RAS servers in your network. The **ContentPackager.jar** file is in the SDK folder.

Important: RAS installed on a Windows server supports both Java and .NET RAS operations. However, RAS installed on a Linux server only supports Java RAS operations, and does not support .NET RAS operations.

Installing the content

The **Content Packager** extracts the packaged libraries and repositories from the **ContentInstaller.jar** file into the OO home directory, in the **updates\content\module\version** folder on the target server. It then updates the Central server repository, local repository, or RAS specified in the arguments passed to it. By default, the Central repository at **https://localhost:8443** is updated, as are all the RASes referenced by it.

To install the content

- In a command window, type:

```
java -jar <name>-contentInstaller.jar [-ep encrypted repository
<password>] [-centralURL url] [-centralUsername <username>]
-centralPassword <password> | -ras RAS URL [-home iconclude_home]
[-repo localRepo]
```

  The options are:

  - `-ep`

    Specifies the encrypted password to the target repository.

  - `-centralURL`

    Specifies the URL of the Central repository to be updated. The default URL is **https://localhost:8443**

  - `-centralUsername`

    Specifies the user name for accessing Central. The default is admin.

  - `-centralPassword`

    Specifies the password for accessing Central.

  - `-ras`

    Specifies the URL of the RAS to be updated. If you don't specify a RAS URL, the content is installed on all the RASes that are registered in the target repository.

- `-home`

  Specifies the path of your OO installation. This defaults to the value of the `ICONCLUDE_HOME` environment variable.

- `-repo`

  Specifies the path of the repository to update. This defaults to the **centralURL** repository.

You can specify a local repository, but as a best practice, update a Central repository and then have the authors who use that Central repository update their local repositories.

For additional information on publishing to, and updating from the public repository, see the *Studio Authoring Guide*.

Example

```
Java –jar ExampleInstalleration-OO9.0-ContentInstaller.jar -centralUrl
https://localhost:8443 –centralUsername admin –centralPassword
password
```

Creating the XML configuration file

The first step in packaging content is to create an XML configuration file. This must contain the information needed to package and install content. It must include the location of the repository that contains the source server content, and the path in the repository or library to be published to Central or a RAS.

The XML configuration file requires the following XML elements:

- A project element that defines the content module properties.
- A RAS element that specifies the location on the source server of the libraries to be updated on the target RAS servers.
- An archive element that provides information on how to and where to install the updated content on the RAS servers.
- A repository element that specifies the location on the source server of the repositories to be updated and where to install them on the target Central servers.

These elements are described in the following sections. For an example, see "XML configuration file example" on page 103.

Using the Content Packager

The packaging content for distribution process involves the following steps:

1. Create an XML configuration file that defines:
   - The content to install on the target Central or RAS servers.
   - The RAS libraries to update on the target RAS servers.
   - The repositories to update on the target Central servers.

To learn how to create the XML configuration file, see "Creating the XML configuration file" on previous page.

2. Package the content.

   The **Content Packager** uses the information in the XML configuration file to incorporate the content into a content module. It then creates a file named **<name>-ContentInstaller.jar** which contains the content module and the classes needed to install it.

   For instructions on using the **Content Packager**, see "Packaging, depackaging, and repackaging content" on page 106.

   The `<name>` is specified in `<project>`.

3. Create the OO home directory folder structure on the target server where you want to install the content. You can skip this step if the target server has Studio installed on it.

   For the necessary folder structure, see "Configuring the OO home directory structure" on page 108.

4. Install the packaged content.

   The **Content Packager** extracts the content into the OO home directory, in the **updates\content\module\version** folder on the target server. It then updates the Central repository, the local repository, or a RAS, depending on the arguments you use when installing the package. For instructions, see "Installing the content" on page 98.

When you update a Central repository, it is important to let the authors who access that repository know that they should update their local repositories as well. For information on updating from a Central repository, see the *Studio Authoring Guide* or the online *Studio Help System*.

The project element

The project element contains information about the content module created from your content by the **Content Packager**.

Syntax

The syntax is:

```
<project schema_version="value" name="value" version="value"
fullname="value"></project>
```

The following table contains the project element attributes and their values:

Attribute	Value
`schema_version`	Always set this attribute to a value of `2`. This attribute is reserved for future use.
`name`	A short name for the content package. The name cannot contain spaces or special characters such as quotation marks (“), asterisks (*), slash marks (/ and \), colons (:), angle brackets (< or >), question marks (?), vertical bars (|), apostrophes (❻), ampersands (&), semicolons (;), and number sign (#).

Attribute	Value
`fullname`	The full display name of the content package. The full name can contain spaces, but no special characters, should be easily understandable and reflect the package contents .
`version`	The content package version number. The version number is unique to the content package, and does not need to be related to the OO version numbering scheme.

Example

```
<project schema_version="2" name="ExampleIntegration-OO"
version="1.0.0" fullname="OO Example Integration Content
Installer"></project>
```

The ras element

The `ras` element defines the RAS libraries (JAR or .NET files) to update on the target RAS servers. The `ras` elements must be nested inside `project` elements.

Syntax

```
<ras type="value" dir=”path” description=”value” ></ras>
```

The following table contains the `ras` element attributes and their values:

Attribute	Value
`type`	The language in which the RAS libraries are written (Java or .NET). Valid values: `java`, `dot_net`
`dir`	The path where the libraries are located on the source server, relative to the path from which you are running the **Content Packager**. The subfolders in this path will be packaged only if they include at least one file.
`description`	The description of the RAS libraries.

Example

```
<ras type="java" dir="../dist/JRAS" description="Example Integration
Java RAS Libraries"> </ras>
```

The archive element

The archive element provides the information needed to install the RAS libraries on the target RAS servers and determine which files to package for installation on the RAS. Each archive element must be nested inside a RAS element.

Syntax

```
<archive isLib="value" [libFolderName=”value”]>path</archive>
```

`path` is the path to the archive relative to the `dir` established in the ras element, using an asterisk (*) as a wildcard character.

The following table describes the `isLib` archive element attribute and value:

Attribute	Attribute value
isLib	Set this attribute to a false value if the library contains IActions. Set it to a true value if the library does not contain IActions.
libFolderName	The folder (under **lib**) where the libraries are installed. If the isLib attribute is set to true, the libFolderName attribute must be specified.

Examples

```
<archive isLib=”false” >ExampleIntegration.jar</archive>
<archive isLib=”true libFolderName="ExampleIntegration">
lib/*.jar</archive>
```

The second example installs the specified libraries into the OO home folder, in the **RAS\Java\Default\repository\lib\ExampleIntegration** folder.

The repository element

The repository element specifies where can **Content Packager** find the repository on the source server and which path to publish to Central. The repository element must be nested inside a project element.

Syntax

```
<repository path="value" desctiption="value">

<include>path</path>

</repository>
```

where <include> path is the source repository directory. The path attribute should be the path for the whole repository directory. Only relative paths are supported.

The following table contains the repository element attributes and their values:

Attribute	Value
path	The source content path within the source repository, relative to the path from which you are running the **Content Packager**. It can also be the path of the target content within the target repository.
description	The description for the repository

Example

The following repository element example instructs the packager to package all repository information under **../dist/Repository/ExampleIntegrationRepo**, and to install or update all of the contents under source /Library to target /Library.

```
<repository path="../dist/Repository/ExampleIntegrationRepo
description=”ExampleIntegraton Respository”>
<include>/Library</include>
</repository>
```

XML configuration file example

The following is an example of the XML configuration file you need to create for use with the **Content Packager**.

XML configuration file example

```
<?xml version="1.0" ?>

<project schema_version="2" name="ExampleIntegration-OO9.0"
version="1.0.0" fullname="OO Example Integration Content Installer">

<ras type="java" dir="../dist/JRAS" description="Example Integration
Java RAS Libraries">

<archive isLib=”false” >ExampleIntegration.jar</archive>

<archive isLib=”true”
libFolderName="ExampleIntegration">lib/*.jar</archive>

</ras>

<ras type="dot_net" dir="../dist/NRAS"description="Example
Integration .NET RAS Libraries">

<archive>*.dll</archive>

</ras>

<repository path="../dist/Repository/ExampleIntegrationRepo”
description=”ExampleIntegraton Repository”>

<include>/Library</include>

</repository>

</project>
```

You can also incorporate the packaging process into an automated build process.

Example of incorporating packaging into an automated build

```
<project name="Sample Ant Buildfile"
         xmlns:oo="antlib:com.hp.oo.content.utilities.packager.ant"
         xmlns="antlib:org.apache.tools.ant">
    <taskdef uri="antlib:com.hp.oo.content.utilities.packager.ant"

resource="com/hp/oo/content/utilities/packager/ant/antlib.xml"
             classpath="/path/to/ContentPackager.jar" />


    <oo:packager outputFolder="${build.dir}/OO-Packages"
                 property="acme.installer"
                 name="Acme"
                 version="${version}"
                 fullname="My Acme 10.0 Integration"
                 description="Example Content Installer">
        <ras type="java"
             dir="${dist.dir}/"
             description="Blah">
            <archive>ACME.jar</archive>
            <archive libFolderName="ACME">ACME-lib.jar</archive>
            <archive libFolderName="ACME">
                <include name="*-commons.jar" />
                <exclude name="*test*.jar" />
            </archive>
        </ras>


        <repository path="${dist.dir}/repo"
                    description="Acme Repository content for ASGARD
OO release">
            <include name="/Library/Integrations/Acme/" />
            <exclude name="/Library/Integrations/Acme/Deprecated/"
/>
        </repository>
    </oo:packager>
```

```
</package>
```

Packaging, depackaging, and repackaging content

The **Content Packager/Depackager** uses the XML configuration file to:

- Package your content into a content module.
- Generate the content installer JAR file which contains the content module and installation classes.
- Depackage the content installer JAR file.
- Repackage the old content and make an installer for a newer version of OO, based on the **ContentPackager.jar** version.
  For example, if you have a content installer for OO 7.51, you cannot use it to install the content on OO 9.00. You can use the repackaging mode to package the content to an installer that works for OO 9.00.

For information on the XML configuration file makeup, see "Creating the XML configuration file" on page 99.

To package the content

- Type the following (in a command window):

```
java  -jar ContentPackager.jar [-mode package] -configFile <file>
[-outputFolder <dir>]
```

The options are:

- `-mode`

  Indicates the action to apply to the contents. The default is to `package`.

- `-outputFolder`

  Specifies the directory where **ContentPackager** will generate the content installer JAR file.

- `-configFile`

  Specifies the path of the XML configuration file.

Example

```
java -jar tools/ContentPakcager.jar -mode package -configFile
config/package.xml -outputFolder output
```

To depackage the content

- Type the following (in a command window):

```
java  -jar ContentPackager.jar -mode depackage -installerJar <file>
[-outputFolder <dir>]
```

The options are:

- `-mode`

  Indicates the action to apply to the contents. For depackaging, use `depackage`.

- `-outputFolder`

  Specifies the directory where the **ContentPackager** will generate the content files.

- `-installerJar`

  Specifies the path of the content installer file.

Example

```
java –jar tools/ContentPakcager.jar –mode depackage –installerJar
output/ExampleIntegration-OO9.0-ContentInstaller.jar –outputFolder
output/depackage
```

To repackage the content

- In a command window, type the following:

  ```
  java  -jar ContentPackager.jar -mode repackage –installerJar <file>
  [-outputFolder <dir>]
  ```

  The options are the following:

  - `-mode`

    Indicates the action to apply to the contents. For repackaging, use `repackage`.

  - `-outputFolder`

    Specifies the directory where the **ContentPackager** will generate the new content installer.

  - `-installerJar`

    Specifies the path of the old content installer file.

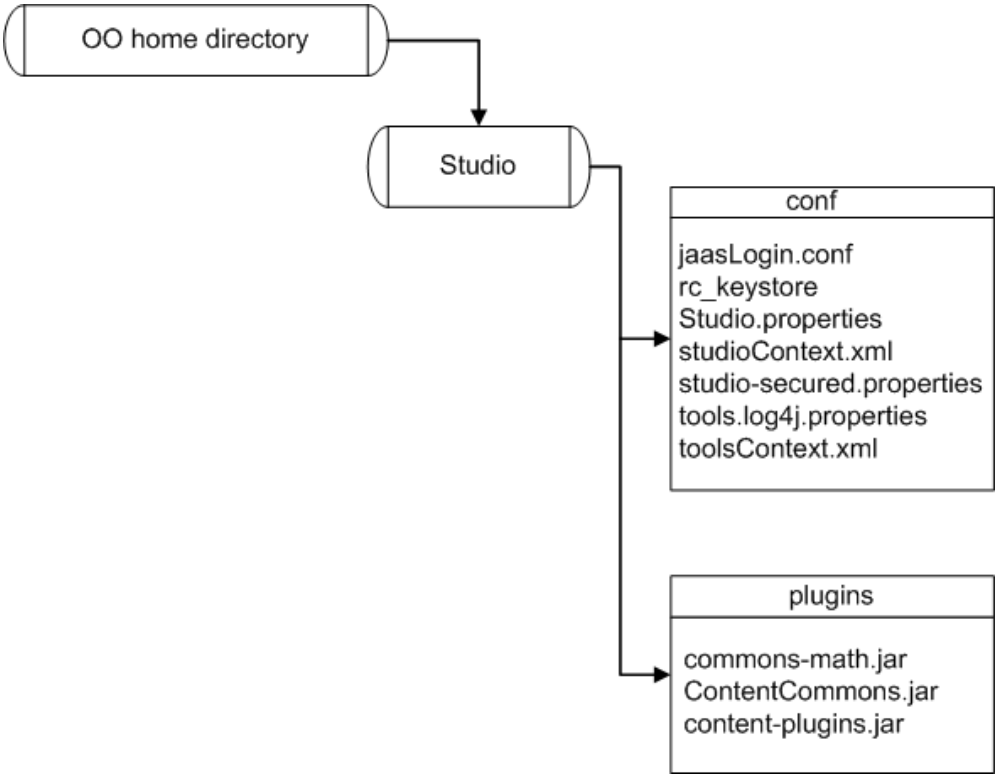
Example

```
java –jar tools/ContentPakcager.jar –mode repackage –installerJar
output/ExampleIntegration-OO7.51-ContentPackager.jar –outputFolder
output/repackage
```

Configuring the OO home directory structure

The OO home directory is the folder where OO is installed. By default, this is **C:\Program Files\Hewlett-Packard\Operations Orchestration**.

Before installing the packaged content, the OO home directory structure must be in place on the target server. If the server has Studio installed on it, the structure is already in place. If not, configure the OO home directory structure on the target server as shown in the following figure. The **conf** and **plugins** folders need to contain all the files from the Studio version that was used to develop the content.



OO home directory structure

After you create the needed folder structure, set the `ICONCLUDE_HOME` operating system environment variable to the OO home directory.

Chapter 6

Inspecting a repository

The **RepoInspector** utility is a tool that inspects a repository. **RepoInspector.jar** can be found in the SDK home directory.

When an inspection fails, a warning message is issued. The message contains valuable information such as:

- The repository UUID and path
- A description of the problem
- The user impact,if any
- Suggested remedies

This information enables content developers of all levels - OO developers and third-party content developers - to understand, diagnose, and fix content problems.

RepoInspector.jar does the following:

- "Checking best practices " below. Perform best practices checks and reports violations.
- "Checking version compatibility" on page 111. Perform version compatibility checks and detects compatibility violations with previously-released snapshots of the same repository
- "Generating release notes " on page 113.
- "Listing repository contents" on page 114.

Checking best practices

RepoInspector checks for best practice violations and writes all encountered violations to a file. It performs the following best practice checks:

- Deprecated Usage

  Checks for flows that use deprecated operations—any operations in a folder named **Deprecated**. Deprecated flows are not subject to this check.

- Description Exists

  Verifies that all objects have descriptions.

- PRE Tags in Description

  Checks that the text in the Studio **Description** tabs is written between `<pre>` and `</pre>` tags. These tags are required to make flow descriptions appear correctly in Central.

- Consistent Input Bindings

  Checks for problems in input bindings, such as user names (or passwords or host names) bound to specific values, unencrypted passwords, and empty prompts.

- Obsolete References

  Flags the terms "JRAS" or "NRAS" in descriptions. Only the term "RAS" is to be used.

- Description Consistency

  Checks that the operation descriptions document all the inputs, responses, and results. This check is required because of a deficiency in the IAction interface used to communicate between infrastructure and content.

- Response Type Consistency

  Verifies that response types (`success`, `diagnosed`, `failure`/`failed`) are consistent with their Java enumeration values (`Type.RESOLVED`, `Type.DIAGNOSED`, `Type.ERROR`).

- Questionable Deprecation

  Finds operations that appear to have been deprecated for no reason. Tthe new version of the operation has no new required inputs and does not lack any outputs that the original operation had.

Syntax

```
Java –jar RepoInspector.jar -repo1 <path to repo1> -bestpractices
[-outputFolder <path>][-includePaths <repo path>][-excludePaths
<repo path>][-exemptionFile <path>]
```

The options are:

- `-repo1 <repository URL>`

  The value of `<repository URL>` can be:

  - Strings in the following form:

    `<user>:<password>:<encPassword>@host:port`

  - Folder paths to local repositories.

    The path has to be relative to where **RepoInspector** is run.

  - The path to a content installer.
  - The path to a local ZIP file.
  - The URL to a remote ZIP file.

- `-bestpractices`

  Specifies to run a best practices check for the repository.

- `-outputFolder`

  Specifies the folder in which to place the Best Practice Violations result files.

- `-includePaths`

  Specifies the repository paths for best practice checking. The repository paths should be in a quoted, comma-delimited list.
  The default is `includePaths: /Library,/Configuration`

- `-excludePaths`

Specifies which repository paths should be excluded for best practice checking. The repository paths should be in a quoted, comma-delimited list.
The default is `excludePaths: /Repository,/Configuration/Groups, /Configuration/System Properties, /Configuration/System Accounts, /Configuration/System Filters`

- `-exemptionFile`

  The path of the properties file of exemptions to test. The format of the properties file is:

  ```
  UUID|Library Path = <comma separated list of tests>
  ```

  The following is an example of an exemption property file:

  ```
  ! /Library/..., done because...

  905742c8-2412-4476-a7d3-49d0acfa4b40 = InputTest

  ! Done because...

  /Library/Integrations/My Integration/Add Stuff =
  DeprecatedUsageTest,InputBindingTest
  ```

The following is a sample command that starts the repository inspector:

```
Java –jar RepoInspector.jar –repo1 ../ExampleIntegrationRepo
–bestPractices
```

Checking version compatibility

RepoInspector can verify compatibility between a new repository and old (released) repository. The purpose is to identify any content changes that can break the customer's flows.
RepoInspector performs the following version compatibility checks:

- Existence

  Every object (UUID) in the old repository must be present in the new repository. If this test fails, all further compatibility checking is skipped.

- Inputs

  Every required input in the new repository must be required in the old repository. Compatibility is broken if you introduce a new required input, or change an existing input, making it required.

- Version

  If an object in the new repository is present in the old one, then it must be based on the latest version of the old one. In other words, if the new repository object is based on an object in the old repository that has been subsequently changed, then it is flagged as a violation. Conversely, if the embedded version numbers of the object have not changed from the old to the new repository yet the objects really have undergone some changes, then it is also flagged. Changes are normally only possible by manually manipulating the repository XML object outside Studio

Syntax

```
Java –jar RepoInspector.jar -repo1 <path to repo1> -repo2 <path to
repo2>  -compatibility [-outputFolder <path>]  [-includePaths <repo
path>] [-excludePaths <repo path>] [-exemptionFile <path>]
```

The options are:

- `-repo1 <repository URL>`

  The value of `<repository URL>` can be:

  - Strings in the following form:

    `<user>:<password>:<encPassword>@host:port`

  - Folder paths to local repositories. The path has to be relative to where **RepoInspector** is run.
  - The path to a content installer.
  - The path to a local ZIP file.
  - The URL to a remote ZIP file.

- `-repo2 <repository URL>`

  The value of `<repository URL>` can be:

  - Strings in the following form:

    `<user>:<password>:<encPassword>@host:port`

  - Folder paths to local repositories. The path has to be relative to where **RepoInspector** is run.
  - The path to a content installer.
  - The path to a local ZIP file.
  - The URL to a remote ZIP file.

- `-compatibility`

  Specifies to run a compatibility check for the repository.

- `-outputFolder`

  The path to the **Compatibility** folder containing the result files.

- `-includePaths`

  Specifies which repository paths to include for the compatibility checking. The repository paths should be in a quoted, comma-delimited list.
  The default is `includePaths: /Library,/Configuration`

- `-excludePaths`

  Specifies which repository paths to exclude for the compatibility checking. The repository paths should be in a quoted, comma-delimited list.
  The default is `excludePaths: "/Repository,/Configuration/Groups, /Configuration/System Properties,/Configuration/System Accounts, /Configuration/System Filters"`

- `-exemptionFile`

  The path of the properties file for exemptions to test.

Example

```
Java –jar RepoInspector.jar –repo1 ../ExampleIntegrationRepo –repo2
../AnotherExampleIntegrationRepo -compatibility
```

Generating release notes

RepoInspector can generate release notes for specified repositories.

Syntax

```
Java –jar RepoInspector.jar -repo1 <path to repo1>  -releasenotes
[-outputFolder <path>]  [-includePaths <repo path>] [-excludePaths
<repo path>]
```

The options are:

- `-repo1 <repository URL>`

  The value of `<repository URL>` can be:

  - Strings in the following form:

    `<user>:<password>:<encPassword>@host:port`

  - Folder paths to local repositories. The path has to be relative to where **RepoInspector** is run.
  - The path to a content installer.
  - The path to a local ZIP file.
  - The URL to a remote ZIP file.

- `-releasenotes`

  Specifies to generate release notes for the repository.

- `-outputFolder`

  The path to the **ReleaseNotes** folder containing the result files.

- `-includePaths`

  Specifies which repository paths to include for the release notes generation. The repository paths should be in a quoted, comma-delimited list.
  The default is `includePaths: /Library,/Configuration`

- `-excludePaths`

  Specifies which repository paths to exclude for the release notes generation. The repository paths should be in a quoted, comma-delimited list.
  The default is `excludePaths: /Repository,/Configuration/Groups, /Configuration/System Properties,/Configuration/System Accounts, /Configuration/System Filters`

Example

```
Java –jar RepoInspector.jar –repo1 ../ExampleIntegrationRepo
–releasenotes
```

Listing repository contents

RepoInspector can list the content of specified repositories.

Syntax

```
Java –jar RepoInspector.jar -repo1 <path to repo1>  -listing
[-outputFolder <path>]  [-includePaths <repo path>] [-excludePaths
<repo path>]
```

The options are:

- `-repo1 <repository URL>`

  The value of `<repository URL>` can be:

  - Strings in the following form:

    `<user>:<password>:<encPassword>@host:port`

  - Folder paths to local repositories. The path has to be relative to where **RepoInspector** is run.
  - The path to a content installer.
  - The path to a local ZIP file.
  - The URL to a remote ZIP file.

- `-listing`

  Specifies to generate a listing of the repository content.

- `-outputFolder`

  The path to the **Listing** folder that contains the result files.

- `-includePaths`

  Specifies which repository paths to include for the repository listing. The repository paths should be in a quoted, comma-delimited list.
  The default is `includePaths: /Library,/Configuration`

- `-excludePaths`

  Specifies which repository paths to exclude for the repository listing. The repository paths should be in a quoted, comma-delimited list.
  The default is `excludePaths: "/Repository,/Configuration/Groups, /Configuration/System Properties,/Configuration/System Accounts, /Configuration/System Filters"`

Example

```
Java –jar RepoInspector.jar –repo1 ../ExampleIntegrationRepo
–listing
```

Chapter 7

Automating flow testing

The **AutoTest** utility is an automated OO content tester with stress testing capabilities. **AutoTest.jar** is found in the SDK home directory. The Central service must be available, as the tool invokes the flows in OO Central.

Syntax

```
java [system properties] -jar AutoTest.jar [parameters] <xmlFilePath>
```

Example

```
Java –Dlogfile=output.log –jar AutoTest.jar –dbpass iconclude
input.xml
```

System properties

Use the following syntax to specify system properties for the **AutoTest** utility:

```
–D<property>=<value>
```

For example:

```
–Dlogfile=output.log
```

The following table describes the properties and the aspects of the system that they regulate.

This property	Affects this
`Logfile`	The path to the output log file.
`stress.conn.timeout`	The connection timeout, in seconds.
`stress.read.timeout`	The read timeout, in seconds.
`-threaddelay`	The delay before each thread begins. Default is `0`.
`-threaddelayevery`	Apply ‘threaddelay’ everything n-th trhead to run. Default is `1`.
`-threads`	The maximum number of concurrent threads. Default is `1`.
`-varxml`	The XML file of variable overrides.
`-xmlfile`	The XML file of central and variable overrides.

Parameters

Use the following syntax to specify parameters for the **AutoTest** utility:

```
–name <value>
```

For example:

```
-dbpass iconclude
```

The parameters are:

- `-dbhost`

  The OO Central database host. The default is the same host as OO central.

- `-dbname`

  The OO Central database name. The default is `dharma`.

- `-dbpass`

  The OO Central database password.

- `-dbport`

  The OO Central database port.

- `-dbtype`

  The OO Central database type. The valid values are `oracle, mysql,` and `mssql`.

- `-dbuser`

  The OO Central database user.

- `-helpxml`

  Displays help information in the format of an XML file.

- `-host`

  The OO Central host.

- `-https`

  Specifies whether to use SSL. The valid values are `true` and `false`. The default is `true`.

- `-user`

  The OO Central username.

- `-pass`

  The OO Central password.

- `-port`

  The OO Central port.

- `-quiet`

  Specifies minimal logging to the console.

- `-runcount`

  The number of times to repeat each runnable element. Default is `1`.

Note: Most of the above options can be specified in an XML input file. The values you specify in a command line take precedence over the values contained in XML files, with the exception of `-xmlfile FILENAME`, which loads an entire XML file of overrides.

Sample XML input files

The following is the example of an all-in-one XML configuration file:

```
<?xml version="1.0"?>
<stress-run>
    <!-- Main config block -->
    <central>
        <!-- Central config -->
        <host>myhost.battleground.ad</host>
        <user>admin</user>
        <pass>secret</pass>
        <https>true</https>
        <port>8443</port>

        <!-- Database config -->
        <dbhost>myhost.battleground.ad</dbhost>
        <dbname>dharma</dbname>
        <dbuser>dharma_user</dbuser>
        <dbpass>secret</dbpass>
        <dbtype>mssql</dbtype>

        <!-- Stress tool config -->
        <runcount>1</runcount>
        <threads>5</threads>
    </central>

    <!-- Definitions -->
    <variables>
        <myhost>myhost.battleground.ad</myhost>
        <myip>192.168.5.55</myip>
```

```
    </variables>


    <!-- Flows -->
    <flow>
        <description>Connectivity Step1 Test</description>
        <name>/Library/Accelerator Packs/
              Network/ConnectivityTest</name>
        <input name="host">${myhost}</input>
        <input name="lossThreshold">5</input>
        <input name="latencyThreshold">300</input>
        <response>success</response>
        <rule>
            <key>FailureMessage</key>
            <value></value>
            <comparator>contains</comparator>
        </rule>
    </flow>
    <flow>
        <description>JRAS Test 01</description>
        <name>/Library/Stress/JRAScommand</name>
        <input name="hostname">${myhost}</input>
        <response>success</response>
    </flow>


    <!-- Operations -->
    <operation>
        <description></description>
        <name>/Library/Operation/Some Operation</name>
        <input name="myhost" />
        <response>success</response>
    </operation>
</stress-run>
```

The following is an example of a separate Central\variables sample XML configuration file.

Main input XML file:

```
<?xml version="1.0"?>

<anything>

    <central>

        <runcount>1</runcount>

        <threads>1</threads>

        <xmlfile>opsforceenv.xml</xmlfile>

    </central>

    <variables>

        <xmlfile>Input Variables.xml</xmlfile>

    </variables>


    <!-- ************************************************ -->

    <!--  Operation Name: Library/Integrations/Hewlett-Packard

    <!--  /Universal CMDB/Add Object -->

    <!-- ************************************************ -->


    <!-- DL1 -->

    <flow>

        <description>Regression Test 01</description>

        <name>/Library/Havok_Regression/Integrations/
           Hewlett-Packard/Universal_CMDB/OP_Add_Object_HVK01</name>

        <input name="cmdbHost">${cmdbHostName}</input>

        <input name="cmdbPort">${cmdbPortValue}</input>

        <input name="username">${cmdbUsername}</input>

        <input name="password">${cmdbPassword}</input>

        <input name="objectType">${AddcmdbObjecttype}</input>

        <input name="prop">${AddcmdbProp}</input>

        <input name="cmdbVersion">${cmdbVersion}</input>

        <response>success</response>

    </flow>
```

```
    <!-- DL2a -->
    <flow>
        <description>Regression Test 02</description>
        <name>/Library/Havok_Regression/Integrations/
           Hewlett-Packard/Universal_CMDB/OP_Add_Object_HVK01</name>
        <input name="cmdbHost">${badcmdbHostName}</input>
        <input name="cmdbPort">${cmdbPortValue}</input>
        <input name="username">${cmdbUsername}</input>
        <input name="password">${cmdbPassword}</input>
        <input name="objectType">${cmdbObjecttype}</input>
        <input name="prop">${cmdbProp}</input>
        <input name="cmdbVersion">${cmdbVersion}</input>
        <response>Failure</response>
    </flow>

    ...
    ...
</stress-run>



Central Xml file:


<?xml version="1.0"?>
<anything>
    <central>
        <host>16.93.12.44</host>
        <port>8443</port>
        <user>admin</user>
        <pass>admin</pass>
        <https>true</https>

        <dbhost>16.93.12.44</dbhost>
        <dbname>oo</dbname>
```

```
        <dbuser>admin</dbuser>
        <dbpass>admin</dbpass>
        <dbtype>mssql</dbtype>
    </central>
</anything>

Variables Xml file;

<?xml version="1.0"?>
<anything>
    <variables>
        <!-- ****************************************** -->
        <!--                uCMDB Variables                -->
        <!-- ****************************************** -->
        <cmdbHostName>15.23.143.2</cmdbHostName>
        <cmdbPortValue>8080</cmdbPortValue>
        <cmdbUsername>admin</cmdbUsername>
        <cmdbPassword>admin</cmdbPassword>
        <cmdbVersion>8</cmdbVersion>

        <!-- /Library/Havok_Regression_Ros/uCMDB_ros/Add_Object_HVK01
-->
        <AddcmdbObjecttype>unix</AddcmdbObjecttype>
        <AddcmdbProp>host_key=Test</AddcmdbProp>

        ...
        ...
    </variables>
</anything>
```

Chapter 8

Debugging OO client/server problems

The communication between OO components is accomplished using SSL (Secure Sockets Layer). SSL encrypts data that is transmitted between clients and servers through the Internet. When a client/server problem occurs with OO, SSL does not allow you to capture the data packets transmitted between the client and the server to validate that data is being sent properly. For example, a call to the Central Web Service that is not working correctly.

The solution is to enable HTTP access, which allows you to capture live data packets and inspect or compare them. This is usually the best way to debug OO client/server problems.

You can use the two following procedures for debugging:

- The first procedure allows HTTP access to Central.
- The second procedure does the same thing for RAS.

To allow HTTP access to Central

1. Stop the **RSCentral** service.
2. In a text editor, open the **applicationContext.xml** file, located in the **Central\WEB-INF** folder of the OO home directory.
3. Comment out every line, in any section, that begins with `<!-- HTTPS_SECTION_BEGIN -->` and ends with `<!-- HTTPS_SECTION_END -->`, and then save the file.
4. Open the **web.xml** file, located in the **Central\WEB-INF** folder of the OO home directory.
5. Comment out every line, in any section, that begins with `<!-- HTTPS_SECTION_BEGIN -->` and ends with `<!-- HTTPS_SECTION_END -->`, and then save the file.
6. Restart the **RSCentral** service.
7. Connect to port `8080` using HTTP, rather than port `8443` using HTTPS.

To allow HTTP access to RAS

1. Stop the **RSJRAS** service.
2. In a text editor, open the **jetty.xml** file, located in the **RAS\Java\Default\webapp\conf** folder of the OO home directory.
3. Comment out the line:

   ```
   <New class="org.mortbay.jetty.security.SslSelectChannelConnector">
   ```

4. Add the following line directly under the line you commented out in the previous step:

   ```
   <New class="org.mortbay.jetty.nio.SelectChannelConnector">
   ```

5. Comment out the lines starting with:

- `<Set name="Keystore">`
- `<Set name="Password">`
- `<Set name="KeyPassword">`
- `<Set name="NeedClientAuth">`

6. Save the file.
7. Open the **applicationContext.xml** file, located in the **RAS\Java\Default\webapp\WEB-INF** folder of the OO home directory.
8. Comment out every line, in any section, that begins with `<!-- HTTPS_SECTION_BEGIN -->` and ends with `<!-- HTTPS_SECTION_END -->`, and then save the file.
9. Open the **web.xml** file, located in the **RAS\Java\Default\webapp\WEB-INF** folder in the OO home directory.
10. Comment out every line, in any section, that begins with `<!-- HTTPS_SECTION_BEGIN -->` and ends with `<!-- HTTPS_SECTION_END -->`, and then save the file.
11. Restart the **RSJRAS** service.
12. Connect to port `9004` using HTTP instead of HTTPS.

To turn off the HTTP connections allowance for either procedure, reverse the procedure.

Chapter 9

OO SOAP API Reference

The SOAP API, primarily documented in the *SOAP API Javadocs*, includes:

- "Configurations" on page 128.
- " Clusters" on page 132.
- "Flows" on page 134.
- "Groups and Users Management" on page 143.
- " Repositories" on page 158.
- " Runs" on page 180.
- "Scheduler" on page 220.
- "Selection lists" on page 271.

SOAP API Availability Chart

The following tables contain the currently available SOAP API methods, classes and information on their timeline:

Method name	Available since
getLWSSOConfig	9.06
updateLWSSOConfig	9.06
getClusterNodes	9.03.001
getFlowDetails	earlier version
getFlowGraph	earlier version
getFlowInputDescriptions	earlier version
moveFlow	9.03.001
createGroup	9.06
updateGroup	9.06
deleteGroup	9.06
createFolder	9.03.001
moveFolder	9.03.001

Method name	Available since
search	earlier version
list	earlier version
getCategories	earlier version
getAttributes	9.04
getPermissions	9.06
setPermission	9.06
renameRepoEntity	9.06
deleteRepoEntity	9.06
updateDescription	9.06
pauseRun	earlier version
resumeRun	earlier version
cancelRun	earlier version
runFlow	earlier version
runFlowEx	earlier version
getFlowRunHistory	earlier version
getFlowRunHistoryByRunId.htm	9.07
getFlowsRunHistory	earlier version
getRunStatus	earlier version
getRunStatusEx	earlier version
getStatusForRuns	earlier version
scheduleFlow	9.03.001
getSchedule	9.03.001
pauseSchedule	9.03.001
isSchedulePaused	9.03.001
resumeSchedule	9.03.001
deleteSchedule	9.03.001
isSchedulerEnabled	earlier version
getScheduledFlows	9.03.001

Method name	Available since
getSchedulesOfFlow	earlier version
getSchedulesForFlowCategory	earlier version
pauseScheduledFlow	9.03.001
isScheduledFlowPaused	earlier version
resumeScheduledFlow	9.03.001
deleteScheduledFlow	9.03.001
createSelectionList	9.03.001
getSelectionList	9.03.001
createUser	9.06
updateUser	9.06
deleteUser	9.06
getUserGroups	9.06

Class name	Available since
WSRunParameters	earlier version
WSRunParametersEx	earlier version
WSRunHandle	earlier version
Serialized form	earlier version
Pair	earlier version
ScheduledFlowInfo	earlier version
ScheduleInfo	earlier version

Constant field values

com.iconclude.*

com.iconclude.dharma.services.wscentralservice.WSCentralService

public static final int	INPUT_LIST_TYPE_ERROR	-1
public static final int	INPUT_LIST_TYPE_FREE_FORM	0
public static final int	INPUT_LIST_TYPE_MULTIPLE_SELECTION	2
public static final int	INPUT_LIST_TYPE_SINGLE_SELECTION	1
public static final int	WS_RUN_HISTORY_MAX_ROWS	100

com.iconclude.dharma.services.wscentralservice.WSInputValueEvaluation

public static final int	BEGINS_WITH_STRING	10
public static final int	CONTAINS_STRING	7
public static final int	ENDS_WITH_STRING	11
public static final int	EQUAL	1
public static final int	EXACT_STRING_MATCH	8
public static final int	GREATER_THAN	5
public static final int	GREATER_THAN_OR_EQUAL	6
public static final int	LESS_THAN	3
public static final int	LESS_THAN_OR_EQUAL	4
public static final int	MATCH_ALL_WORDS	12
public static final int	MATCH_ALWAYS	16
public static final int	MATCH_AT_LEAST_ONE_WORD	13
public static final int	MATCH_NONE_OF_WORDS	14
public static final int	MATCH_ONLY_ONE_WORD	15
public static final int	NOT_EQUAL	2
public static final int	NOT_EXACT_STRING_MATCH	9
public static final int	REGEX_AWK	19
public static final int	REGEX_GLOB	20

public static final int	REGEX_JAVA	17
public static final int	REGEX_PERL5	18
public static final int	SCRIPTLET	21

com.iconclude.dharma.services.wscentralservice.WSInputValueType

public static final int	CREDENTIALS_VALUE_TYPE	4
public static final int	LIST_VALUE_TYPE	2
public static final int	STRING_VALUE_TYPE	1

com.iconclude.dharma.services.wscentralservice.WSScheduleType

public static final int	TYPE_CRON	2
public static final int	TYPE_INTERVAL	1
public static final int	TYPE_UNKNOWN	0

com.iconclude.dharma.services.wscentralservice.WSScheduleUnitType

public static final int	UNIT_HOURS	2
public static final int	UNIT_MINUTES	1
public static final int	UNIT_UNKNOWN	0

Configurations

The methods documented in this section enable you to handle configurations:

- "getLWSSOConfig" below.
- "updateLWSSOConfig" on page 130.

getLWSSOConfig

This method retrieves the LWSSO configuration from the database:

```
WSLwssoConfiguration getLwssoConfig()
```

```
WSLwssoConfiguration{

    Boolean enabled;

    String initString;

    String domain;

    String[] protectedDomains;

}
```

WSLWSSOConfiguration

Field	Type	Description	Comments
enabled	Boolean	A flag indicating whether LWSSO is enabled	Can be `true` or `false`.
initString	String	LWSSO `initString` or passphrase.	A string at least 12 characters long.
domain	String	The OO central server's domain.	Optional - can be empty.
protectedDomains	String[]	An array of strings containing the protected domains.	Optional - can be empty.

Outputs

Output	Type	Description
Result	WSLwssoConfiguration	The `WSLwssoConfiguration` element, representing the LWSSO configuration of the Central server.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- The authenticated user does not have `MANAGE_CONF` or `HEADLESS_FLOWS` capabilites.
- The configuration cannot be retrieved. For example, DB error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getLWSSOConfig soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getLwssoConfigResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getLwssoConfigReturn xsi:type="ns2:WSLwssoConfiguration"
xmlns:ns2="https://localhost:8443/PAS/services/soap/WSCentralService">
            <enabled xsi:type="soapenc:boolean"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">true</enabled>
            <initString xsi:type="xsd:string">CENTRAL_LWSSO_passphrase</initString>
            <domain xsi:type="xsd:string">ssorealm.com</domain>
            <protectedDomains xsi:type="xsd:string" xsi:nil="true"/>
```

```
        </getLwssoConfigReturn>
      </ns1:getLwssoConfigResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

updateLWSSOConfig

This method:

- Updates the LWSSO configuration in the database.
- Reloads the LWSSO context to use the new configuration.

The method is:

```
updateLwssoConfig(WSLwssoConfiguration config)
```

```
WSLwssoConfiguration {

    Boolean enabled;

    String initString;

    String domain;

    String[] protectedDomains;

}
```

WSLWSSOConfiguration

Field	Type	Description	Comments
enabled	Boolean	A flag indicating whether LWSSO is enabled.	Can be `true` or `false`.
initString	String	LWSSO `initString` or passphrase.	A string at least 12 characters long.
domain	String	The OO central server's domain.	Optional - can be empty.
protectedDomains	String[]	An array of strings containing the protected domains.	Optional - can be empty.

Inputs

Input	Type	Description
conf	WSLwssoConfiguration	The `WSLwssoConfiguration` element, representing the new LWSSO configuration of the Central server.

Note that:

- In a clustered environment, after a successful update of the LWSSO configuration, all Central nodes reload the LWSSO context from the database after a period of time. The period of time is defined by the `dharma.lwsso.auto.refresh` property from Central.properties.
- The `enabled` flag is mandatory. If it is not specified, an error is thrown.

- In order to leave unchanged settings from the LWSSO configuration, the inputs must be missing from the request. For example, in order to update only the LWSSO passphrase and leave the rest unchanged, the request must contain only the `enabled` and `initString` inputs.
- The `protectedDomains` and `domain` inputs can be empty in the request.
- If present in the request, the `initString` input must be at least 12 characters long.
- The LWSSO configuration is retrieved from the database, not from the server context. This also happens for cluster environments.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- The authenticated user does not have `MANAGE_CONF` or `HEADLESS_FLOWS` capabilites.
- The configuration cannot be retrieved. For example, DB error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:updateLwssoConfig
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <conf xsi:type="cli:WSLwssoConfiguration"
xmlns:cli="http://client.wscentralservice.services.dharma.iconclude.com">
            <enabled xsi:type="xsd:boolean">true</enabled>
            <initString xsi:type="xsd:string">this_is_init_string</initString>
            <domain xsi:type="xsd:string">this is the domain</domain>
            <protectedDomains xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string
[]" xmlns:wsc="https://16.77.58.113:8443/PAS/services/WSCentralService">
               <protectedDomain>domain1</protectedDomain>
               <protectedDomain>domain2</protectedDomain>
          </protectedDomains>
        </conf>
      </wsc:updateLwssoConfig>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:updateLwssoConfigResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

Clusters

The method documented in this section enable you to handle OO clusters:

- "getClusterNodes" below.

getClusterNodes

This method retrieves information about the cluster's member nodes:

```
WSClusterNodeDetails[] getClusterNodes()
```

```
public class WSClusterNodeDetails implements java.io.Serializable{
    String url;

    String state;

    Long studioSessions;

    Long centralSessions;

    Long activeRuns;

    Long pendingRuns;

}
```

The method returns an array, containing information on each cluster member, including:

- The URL (in the `url` array element).
- The state (in the `state` array element).
- The number of Studio sessions (in the `studioSessions` array element).
- The number of Central sessions (in the `centralSessions` array element).
- The number of active runs (in the `activeRuns` array element).
- The number of pending runs (in the `pendingRuns` array element).

The method is located in the `WSCentralService` class. The method is exposed in the **WSCentralService.wsdl** file.

Inputs

No input is necessary. All cluster members are connected.

Outputs

Output	Type	Description
Result	WSClusterNodeDetails []	A `WSClusterNodeDetails` array containing information about each cluster member (`WSClusterNodeDetails[]`).

Exceptions

An exception is thrown if the user does not have appropriate permissions.

Example:

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getClusterNodes soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getClusterNodesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getClusterNodesReturn soapenc:arrayType="xsd:anyType[1]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getClusterNodesReturn xsi:type="ns2:WSClusterNodeDetails"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
               <url xsi:type="xsd:string">https://host1:8443</url>
               <state xsi:type="xsd:string">online</state>
               <studioSessions xsi:type="xsd:long">3</studioSessions>
               <centralSessions xsi:type="xsd:long">2</centralSessions>
               <activeRuns xsi:type="xsd:long">10</activeRuns>
               <pendingRuns xsi:type="xsd:long">2</pendingRuns>
            </getClusterNodesReturn>
            <getClusterNodesReturn xsi:type="ns2:WSClusterNodeDetails"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
               <url xsi:type="xsd:string">https://host2:8443</url>
               <state xsi:type="xsd:string">online</state>
               <studioSessions xsi:type="xsd:long">2</studioSessions>
               <centralSessions xsi:type="xsd:long">1</centralSessions>
               <activeRuns xsi:type="xsd:long">8</activeRuns>
               <pendingRuns xsi:type="xsd:long">1</pendingRuns>
            </getClusterNodesReturn>
         </getClusterNodesReturn>
      </ns1:getClusterNodesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Flows

The methods documented in this section enable you to handle flows:

- "getFlowDetails" below.
- "getFlowGraph" on page 136.
- "getFlowInputDescriptions" on page 138.
- "moveFlow" on page 142.

getFlowDetails

This method retrieves information about a flow:

```
WSFlowDetails[] getFlowDetails(String[] flowUuids)
```

```
WSFlowDetails {
       String flowDescription
       String[] flowCategories
       String flowDomain
       String lastModifiedBy
       Long lastModifiedTime
       Boolean userCanRunFlow
       String flowVersion
       WSFlowInputDef[] flowParameters
}
```

```
WSFlowInputDef {
       String name
       String value
       String description
       String[] defaultValues
       Boolean bRequired
       Boolean isEncrypted
       Boolean list
       int listType
}
```

Inputs

Input	Type	Description
flowUuids	String[]	An array containing the UUIDs of the flows for which to get details.

Outputs

Output	Type	Description

Result	WSFlowDetails []	The following details for each flow provided through the UUIDs list: • The description of the flow. • The categories to which the flow belongs. • The domain name of the flow. • The user that last modified the flow. • The time of the last modification since 1/1/1970, in milliseconds. • Whether the user calling this method has run permissions on the flow. • The flow version.

Exceptions

`AxisFault` is thrown if:

- `runHandle` is null.
- The provided `runID` is not valid
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getFlowDetails soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuids xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string[]"
xmlns:wsc="https://host:8443/PAS/services/WSCentralService">
            <uuid>31ea143f-2ae5-41bf-b44b-7a8c423011c5</uuid>
         </flowUuids>
      </wsc:getFlowDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getFlowDetailsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getFlowDetailsReturn soapenc:arrayType="ns2:WSFlowDetails[1]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
```

```
            <getFlowDetailsReturn xsi:type="ns2:WSFlowDetails">
               <UUID xsi:type="soapenc:string">31ea143f-2ae5-41bf-b44b-7a8c423011c5</UUID>
               <categories xsi:type="soapenc:Array" xsi:nil="true"/>
               <description xsi:type="soapenc:string">This flow will find services on the
localhost which are not running and restart the one you choose.

&lt;pre>
Responses:
success -  if the services from the list of stopped services is restarted.
failure - if one of the services from the list of stopped services cannot be restarted or
something went wrong.
&lt;/pre></description>
               <domain xsi:type="soapenc:string">Configuration Item</domain>
               <flowVersion xsi:type="soapenc:string">23</flowVersion>
               <lastModifiedBy xsi:type="soapenc:string">admin</lastModifiedBy>
               <lastModifiedTime xsi:type="xsd:long">1274133989949</lastModifiedTime>
               <name xsi:type="soapenc:string">Restart Service - Tutorial Flow</name>
               <parameters soapenc:arrayType="ns2:WSFlowInputDef[1]" xsi:type="soapenc:Array">
                  <parameters xsi:type="ns2:WSFlowInputDef">
                     <defaultValue xsi:type="soapenc:string">localhost</defaultValue>
                     <defaultValues xsi:type="soapenc:Array" xsi:nil="true"/>
                     <description xsi:type="soapenc:string"/>
                     <encrypted xsi:type="xsd:boolean">false</encrypted>
                     <list xsi:type="xsd:boolean">false</list>
                     <listType xsi:type="xsd:int">-1</listType>
                     <name xsi:type="soapenc:string">host</name>
                     <required xsi:type="xsd:boolean">true</required>
                  </parameters>
               </parameters>
               <path xsi:type="soapenc:string">/Library/Tutorials</path>
               <userCanRunFlow xsi:type="xsd:boolean">true</userCanRunFlow>
            </getFlowDetailsReturn>
         </getFlowDetailsReturn>
      </ns1:getFlowDetailsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getFlowGraph

This method retrieves the binary data and the file information from the graph of a flow:

```
WSFlowGraph getFlowGraph(String uuid)
```

```
WSFlowGraph {
       String imageName
       String imageType
       long imageSize
       byte[] imageBinaryData

}
```

- `imageName` is the name of the graph image file.
- `imageType` is the format of the image. Currently, it is PNG.
- `imageSize` is the size of the graph image file.
- `imageBinaryData` is the image byte representation.

Inputs

Input	Type	Description
uuid	String	The flow identifier.

Outputs

Output	Type	Description
Result	WSFlowGraph	The binary data and information about the graph image.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getFlowGraph soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <uuid xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">7107fd7d-5964-4016-a4a8-
9f412df96c7e</uuid>
      </wsc:getFlowGraph>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getFlowGraphResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getFlowGraphReturn xsi:type="ns2:WSFlowGraph"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <imageBinaryData xsi:type="soapenc:base64Binary"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">iVBORw0KGgoAAAANSUhEUgAAAgEAAADICAIA-
AAD+/KVjAAAvl0lEQVR42u2dB1gU1/e/L6BiSO2BriFiF4zdGEus2BAriCAISO-
+9qCC9N7vzP7MDm8kCKzGG1d//8z7n8Zmd3R3W3XvOe+/cKYwDAADw/ysMXwEAAMABAAAA4AAAAABwAAAAADgAAAAHAAA-
AAAOAAAAAAcAAACAAwAAAMABAAAA4AAAAABwAAAADgAAAAHAAAAAAOAAAAAAcAAACAAwAAAMABAAAA4AAAAABwAAAAD-
gAAAAHAAAAAAOAAAAAAcAAACAAwAAAMABAAAA4AAAAABwAAAADgAAAAHAAAAAAOAAAAAAcAAACAAwAAAA4AAAAABwAA-
AIADAAAAwAEAAADgAAAAHAAAAAAOAAAAAAcAAAAA4AAAAABwAAAIADAAAAwAEAAADgAAAAHAAAAAAOAAAAAAcAAAAA-
4AAAAABwAAAIADAAAAwAEAAADgAAAAHAAAAAAOAAAAAAcAAAAA4AAAAABwAAAIADAAAAwAEAAADgAAAAHAAAAAAOAAA-
AAAcAAAAcAAAAAA4AAAAABwAAAADgAAAAHAAAAgAMAAADAAQAAAOAAAAAAcAAAAAA4AAAAABwAAAADgAAAAHAAAAgA-
MAAADAAQAAAOAAAAAAcAAAAAA4AAAAABwAAAADgAAAAHAAAAgAMAAADAAQAAAOAAAAAAcAAAAAA4AAAAABwAAAADgAA-
AAAHAAAAgAMAAADAAQAAAOAAAACAAwAAAMABANSNxy8fB1QEnCw65ZT32+bcrZtytm7M2eKQzYd91ha7rM12mZttMzdvyN-
hsk7HJ5uGm9Q83rUun2Lg2baN12sY1DygcrO47rKa457Dqnv3KVD4sU+xXpNitSLazSLYzT6KwXZ5ka5ZouyxhA4Vp/AYT-
PmyM42yMYm0MKWLWG8Ss16eIXq8XvU43ap1xtL190n7XbM/SZ+X4mQCAA8CbJ/xx+O78PVvytlH1f6sEsDSSD53ItUsi1h-
```

```
pHO7jn+uPHAnXn6dOnkZGR7u7uLi4uF94mXF1d79y5k56e/uLFCzgAKBjfCr+tedvecgFQLA5fqx1u7ZJ5Cz8ZqAsZGRlX-
Jfj4+AQFBYWGhoaFhYVLiIiIIDdERUVFR0fHxMTExsbGxcXFx8cnJCQkSkiWkJKSkpqaeu/evfv37z948CAtLY2q9sOHD2-
nLmZmZWRKys7NzcnJyc3Pz8vLy8/MLCgoKCwuLioqKi4tLJJSWlpaVlZWXl1dUVDx69Ojx48dPnjyyhh7QR0oCHhwethwOA-
IkcAUgHszPrtYqKLb4Tf25Mq1I/Lf1R4KzdgVawTCUA7zHpRmLVbjh9+OPBKAVBf28vLKzg4OCQk5K0SALXqZ8+ePX/+nA-
YB9CdIA/QCOAAogMcvH0t3Af2ediggLOCtTZXSp+U7U46TABaGWhtE2BU9LcXPB+TsAqLu/9svAOLly5eCBmglHADqm4CK-
AOkI4G0WgJAqZc8qLGMcF4auWRCy5kIG9giBWqGmSw54JwRA0IIwNwAHgPrmZNEpYQ7gYqLLO5Eq/nnhJID5IWvWxe7Fzw-
dqw93d3cfH510RAK2hLfv6+sIBoL6RHgb6ts0B1JYqeRWFJID5wWuWhNrg5wO14eLi8gYngcUNW2jbQsN+U62a1tOLXV1d-
4QBQ30iPAnonBEDr6VkSwLxgq7lBVvj5QG1cuHDh3wvAO8bb8bbjovOLvjn6zUcHPhq0d9CgPYM+3Pfht4e/1XbW3nFzh0+UD7V2atv/vlXTa+gzw
 </imageBinaryData>
            <imageName xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">7107fd7d-5964-4016-a4a8-
9f412df96c7e_1349086145237.png</imageName>
            <imageSize xsi:type="xsd:long">12240</imageSize>
            <imageType xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">PNG</imageType>
         </getFlowGraphReturn>
      </ns1:getFlowGraphResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getFlowInputDescriptions

This methods retrieves detailed information about the inputs of a flow:

```
WSInputDescriptor[][] getFlowInputDescriptions(String[] flowUuids)
```

```
WSInputDescriptor {
       String operationUuid
       String errorMessage
       boolean error
       String inputName
       boolean required
       boolean encrypted
       WSInputValueType valueType
       WSInputValueAssignment valueAssignmentPolicy
       WSInputValueEvaluation valueValidationPolicy
       WSInputValueReporting valueReportingPolicy
}

WSInputValueType {
       Int typeId
       String listDelimiter
}

WSInputValueAssignment {
       boolean preAssignFromFlowVariable
       boolean postAssignToFlowVariable
       boolean assignFromConstantValue
       boolean assignFromUserInput
       boolean assignFromSelectionList
       boolean assignFromDomainTerm
       boolean assignFromFlowVariableList
       boolean assignFromSystemAccount
       boolean assignFromLoginCredentials
       boolean assignFromPreviousStepResult
       boolean multipleSelectionAllowed
       String preAssignFlowVariableName
       String postAssignFlowVariableName
       String constantValue
       String userInputPrompt
```

```
        String systemAccountName
        String selectionListName
        String selectionListDescription
        String[] selectionListValues
        String domainTermName
        String domainTermDescription
        String[] domainTermValues
        String flowVariableListName
        String flowVariableListDelimiter
}

WSInputValueEvaluation {
        String evaluatorName
        String evaluatorDescription
        String evaluatorText
        int evaluatorType
        boolean ignoreCase
        boolean regexMatchWholeInput
        boolean regexMultiLine
}

WSInputValueReporting {
        boolean recordInHistory
        boolean recordAsDomainTerm
        String domainTermName
}
```

Note that:

- `operationUuid` is the UUID of the flow containing the input.
- `error` is `true` if an error occurred during the method call.
- `errorMessage` contains the message returned if an error occurred.
- `inputName` contains the name of the input.
- `required` specifies whether the input is required. Valid values: `true`, `false`.
- `encrypted` specifies whether the input is encrypted. Valid values: `true`, `false`.
- `valueType` represents the input type:
  - single text: 1
  - list: 2
  - credential: 4
- If the input type is a list of values, use the list delimiter.
- The assignment details for the input:
  - The input value is assigned from a pre-set flow variable.
  - The input value will be assigned to a flow variable.
  - The input value is assigned from a constant value.
  - The input value is to be user prompted.
  - The input value is assigned from a selection list.
  - The input value is assigned from a domain term.
  - The input value is assigned from a variable list.

- The input value is assigned from a system account.
- The input value is assigned from the credentials of the logged user.
- The input value is assigned from a previous step result.
- If multi selection is allowed for this input.
- The name of the flow variable from which to assign this input.
- The name of the flow variable to be assigned from this input.
- The constant value.
- The message used to prompt the user.
- The name of the system account to assign to the input.
- The name of the selection list to assign to the input.
- The description of the selection list to assign to the input.
- The values of the selection list to assign to the input.
- The name of the domain term to assign to the input.
- The description of the domain term to assign to the input.
- The values of the domain term to assign to the input.
- The name of a flow variable which contains a list of values from which the input can assign its own value.
- The delimiter of the flow variable containing the previously mentioned list of values.

- The way the input value is stored in the database:
  - The input is to be stored in the database for normal reporting. This means that the value is visible in the `Bound values` column of the reports.
  - The input is also going to be stored as a domain term alias. This enables building dashboards.
  - The name of the used domain term, if the input is to be recorded as a domain term.

Inputs

Input	Type	Description
flowUuids	String[]	An array containing the flow UUIDs.

Outputs

Output	Type	Description
Result	WSInputDescriptor[][]	An array of arrays, containing details for the inputs of each flow provided through the UUIDs list.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getFlowInputDescriptions
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuids xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string[]"
xmlns:wsc="https://localhost:8443/PAS/services/WSCentralService">
<uuid>3541d63f-603a-449b-9d43-8e57d7d61482</uuid>
       </flowUuids>
      </wsc:getFlowInputDescriptions>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getFlowInputDescriptionsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getFlowInputDescriptionsReturn soapenc:arrayType="ns2:WSInputDescriptor[][1]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getFlowInputDescriptionsReturn soapenc:arrayType="ns2:WSInputDescriptor[1]"
xsi:type="soapenc:Array">
               <getFlowInputDescriptionsReturn xsi:type="ns2:WSInputDescriptor">
                  <encrypted xsi:type="xsd:boolean">false</encrypted>
                  <error xsi:type="xsd:boolean">true</error>
                  <errorMessage xsi:type="soapenc:string">path not found: '3d1efdc63-5b42-
472f-9d3e-1305c5d2d80'</errorMessage>
                  <inputName xsi:type="soapenc:string" xsi:nil="true"/>
                  <operationUuid xsi:type="soapenc:string">3d1efdc63-5b42-472f-9d3e-
1305c5d2d80</operationUuid>
                  <required xsi:type="xsd:boolean">false</required>
                  <valueAssignmentPolicy xsi:type="ns2:WSInputValueAssignment"
xsi:nil="true"/>
                  <valueReportingPolicy xsi:type="ns2:WSInputValueReporting" xsi:nil="true"/>
                  <valueType xsi:type="ns2:WSInputValueType" xsi:nil="true"/>
                  <valueValidationPolicy xsi:type="ns2:WSInputValueEvaluation"
xsi:nil="true"/>
               </getFlowInputDescriptionsReturn>
            </getFlowInputDescriptionsReturn>
         </getFlowInputDescriptionsReturn>
      </ns1:getFlowInputDescriptionsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

moveFlow

This method moves a flow:

```
String moveFlow(String identifier, String destinationPath, Boolean overwrite)
```

Inputs

Input	Type	Description
identifier	String	The source path (including the name) or the UUID of the flow to move.
destinationPath	String	The location where to move the flow.
overwrite	Boolean	If there is a flow having the same name at the `destinationPath` location, this input specifies whether to add the moved flow as a copy of the existing one. For example, if the name of the flow is **Flow**, the moved flow will be named **Flow (Copy 1)**.

Outputs

Output	Type	Description
Result	String	The path to the moved flow.

Exceptions

Exceptions are thrown if:

- The `identifier` input value does not refer to a flow or an operation.
- The destination contains a flow or an operation having the same name as the flow being moved and the value of the `overwrite` input is `false`.
- The destination contains an object (other than flow or operation) having the same name as the flow being moved.
- The user does not have appropriate permissions.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:moveFlow soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <identifier xsi:type="soapenc:string" xs:type="type:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/2000/XMLSchema-instance">3541d63f-603a-449b-9d43-
8e57d7d61482</identifier>
         <destinationPath xsi:type="soapenc:string" xs:type="type:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/2000/XMLSchema-instance">/Library/My ops Flows</destinationPath>
```

```
         <overwrite xsi:type="soapenc:boolean" xs:type="type:boolean"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/2000/XMLSchema-instance">false</overwrite>
      </wsc:moveFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:moveFlowResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <moveFlowReturn xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/My Ops Flows/Windows Health
Check</moveFlowReturn>
      </ns1:moveFlowResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Groups and Users Management

The methods documented in this section enable you to handle groups and users:

- "createGroup" below.
- "updateGroup" on page 146.
- "deleteGroup" on page 148.
- "createUser" on page 150.
- "updateUser" on page 152.
- "deleteUser" on page 154.
- "getUserGroups" on page 155.

createGroup

This method adds a new group, providing the:

- Group name.
- Description.
- List of capabilities assigned to the group.
- List of external groups to which the group is mapped.

The method is:

```
createGroup(WSGroup group)
```

```
WSGroup {
       String groupName;

       String description;
```

```
        Long capabilities;

        String[] externalGroups;
}
```

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_GROUPS` capabilities.

This method uses the SOAP protocol, sending data in an XML format. Therefore, some special characters are not supported in the string fields.

The character set not being supported includes:

- the non-printable characters.
- `&`, `>`, `<`, `"`, `'`

If you want to use these characters as part of the mentioned fields, replace:

- " with `&quot;`
- ' with `&apos;`
- < with `<`
- > with `>`
- & with `&`

WSGroup

Field	Type	Description
groupName	String	The name of the group being created.
description	String	The description of the group being created. You can leave this field empty.
capabilitiesMask	Int	A bit mask for the existing capabilities. The values are described in the following table. If you leave this field empty, the group is created by default, with no assigned capabilities.
externalGroups	String []	A list of external groups to which to map this group. You can leave this list empty. If an external group contains "," replace it with "_".

Capability	Bit mask
NONE	0
MANAGE_USERS	1
MANAGE_GROUPS	2

Capability	Bit mask
AUTHOR	4
SCHEDULE	8
MANAGE_RUNS	16
RUN_REPORTS	32
MANAGE_CONF	64
VIEW_SCHEDULES	124
HEADLESS_FLOWS	256

Bitwise `OR` between individual capabilities is used in order to combine capabilities. For example, `12 = 4 | 8` represents a valid capabilities mask for `AUTHOR` and `SCHEDULE` capabilities.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_GROUPS` capability.
- The group name is empty.
- The group name already exists.
- The capabilities mask does not point to a valid capability or to a valid capabilities list.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:createGroup soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <group xsi:type="cli:WSGroup"
xmlns:cli="http://client.wscentralservice.services.dharma.iconclude.com">
            <groupName xsi:type="soapenc:string">newGroup</groupName>
            <description xsi:type="soapenc:string">This is a new group</description>
            <capabilities xsi:type="xsd:long">57</capabilities>
            <externalGroups xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string[]"
xmlns:wsc="http://localhost:8086/PAS/services/WSCentralService">
        <eg>external_group</eg>
        </externalGroups>
         </group>
      </wsc:createGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:createGroupResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

updateGroup

This method edits a group with the provided:

- Description.
- List of capabilities assigned to the group.
- List of external groups to which the group is mapped.

The method is:

```
updateGroup(WSGroup group)
```

```
WSGroup{
       String groupName;

       String description;

       Long capabilities;

       String[] externalGroups;
}
```

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_GROUPS` capabilities.

This method uses the SOAP protocol, sending data in an XML format. Therefore, some special characters are not supported in the string field.

The character set not being supported includes:

- the non-printable characters.
- `&`, `>`, `<`, `"`, `'`

If you want to use these characters as part of the mentioned fields, replace:

- " with `&quot;`
- ' with `&apos;`
- < with `<`
- > with `>`
- & with `&`

WSGroup

Field	Type	Description
groupName	String	The name of the group being edited. The group name must exist.
description	String	The description of the group. You can leave this field empty.
capabilitiesMask	Int	A bit mask for the existing capabilities. The values are described in the following table. If you leave this field empty, the group is created by default, with no assigned capabilities.
externalGroups	String []	A list of external groups to which to map this group. You can leave this list empty. If an external group contains "," replace it with "_".

Capability	Bit mask
NONE	0
MANAGE_USERS	1
MANAGE_GROUPS	2
AUTHOR	4
SCHEDULE	8
MANAGE_RUNS	16
RUN_REPORTS	32
MANAGE_CONF	64
VIEW_SCHEDULES	124
HEADLESS_FLOWS	256

Bitwise `OR` between individual capabilities is used in order to combine capabilities. For example, `12 = 4 | 8` represents a valid capabilities mask for `AUTHOR` and `SCHEDULE` capabilities.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_GROUPS` capability.
- The group name is empty.
- The group name does not exist.
- The group name refers to built-in groups.
- The capabilities mask does not point to a valid capability or to a valid capabilities list.

Note: The built in groups (`ADMINISTRATOR`, `AUDITOR`, `EVERYBODY`, `PROMOTER`) cannot be edited.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:updateGroup soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <group xsi:type="cli:WSGroup"
xmlns:cli="http://client.wscentralservice.services.dharma.iconclude.com">
            <groupName xsi:type="soapenc:string">newGroup</groupName>
            <description xsi:type="soapenc:string"></description>
            <capabilities xsi:type="xsd:long">88</capabilities>
            <externalGroups xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string[]"
xmlns:wsc="http://localhost:8086/PAS/services/WSCentralService"><a></a></externalGroups>
         </group>
      </wsc:updateGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:updateGroupResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

deleteGroup

This method deletes a group, providing the group name as the unique group identifier:

```
Boolean deleteGroup(String)
```

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_GROUPS` capabilities.

This method uses the SOAP protocol, sending data in an XML format. Therefore, some special characters are not supported in the string fields.

The character set not being supported includes:

- the non-printable characters.
- `&`, `>`, `<`, `"`, `'`

If you want to use these characters as part of the mentioned fields, replace:

- " with `&quot;`
- ' with `&apos;`
- < with `<`
- > with `>`
- & with `&`

Inputs

Field	Type	Description
groupName	String	The name of the group to delete.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_GROUPS` capability.
- The group name is empty.
- The group name does not exist.
- The group name refers to built-in groups.
- The capabilities mask does not point to a valid capability or to a valid capabilities list.

Note: The built in groups (`ADMINISTRATOR`, `AUDITOR`, `EVERYBODY`, `PROMOTER`) cannot be edited.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:deleteGroup soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <groupName xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">newGroup</groupName>
      </wsc:deleteGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:deleteGroupResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
```

```
        <deleteGroupReturn xsi:type="xsd:boolean">true</deleteGroupReturn>
      </ns1:deleteGroupResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

createUser

This method creates a user account:

```
createUser(WSUser user)
```

```
WSUser {

    String username;

    String password;

    Boolean isInternal;

    Boolean isEnabled;

String[] groups;
}
```

Caution: The create and update functionality handles sensitive data (`password`). The user needs to be aware that on non secure connections (HTTP), this data can be intercepted.

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_USERS` capabilities.

This method uses the SOAP protocol, sending data in an XML format. Therefore, some special characters are not supported in the string fields (`username`, `password`, `groups`).

The character set not being supported includes:

- the non-printable characters.
- `&`, `>`, `<`, `"`, `'`

If you want to use these characters as part of the mentioned fields, replace:

- " with `&quot;`
- ' with `&apos;`
- < with `<`
- > with `>`
- & with `&`

WSUser

Field	Type	Description
username	String	The user name being created. This is case sensitive.

password	String	The password for the user being created.
isInternal	Boolean	An attribute describing whether the user is internal. `false` means that the user is external. If you leave this input empty, it is set to `false` by default, as it is the default behavior in Central UI.
isEnabled	Boolean	An attribute describing whether the user will be enabled. If you leave this input empty, it is set to `true` by default, as it is the default behavior in Central UI.
groups	String[]	A list of the group names to which the user will belong. If you leave this input empty, the user will belong to the `EVERYBODY` group, as it is the default behavior in Central UI. The group names are not case sensitive.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_USERS` capability.
- The user name is empty.
- The user name already exists.
- The password, if provided, has less than 6 characters.
- The password is empty and `isInternal` is `true`.
- The password is not empty and `isInternal` is `false`.
- The list of groups contains group names that do not exist.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:wsc="https://host:8443/PAS/services/WSCentralService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:createUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <user xsi:type="wsc:WSUser">
            <username xsi:type="soapenc:string">user</username>
            <password xsi:type="soapenc:string">password</password>
            <isInternal xsi:type="soapenc:boolean">true</isInternal>
            <isEnabled xsi:type="soapenc:boolean">true</isEnabled>
            <groups xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string[]">
               <g>LEVEL_ONE</g>
               <g>LEVEL_TWO</g>
          </groups>
         </user>
```

```
      </wsc:createUser>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:createUserResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

updateUser

This method updates an existing user account:

```
updateUser(WSUser user)
```

```
WSUser{

    String username;

    String password;

    Boolean isInternal;

    Boolean isEnabled;

    String[] groups;
}
```

Caution: The create and update functionality handles sensitive data (`password`). The user needs to be aware that on non secure connections (HTTP), this data can be intercepted.

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_USERS` capabilities.

This method uses the SOAP protocol, sending data in an XML format. Therefore, some special characters are not supported in the string fields (`username`, `password`, `groups`).

The character set not being supported includes:

- the non-printable characters.
- &, >, <, ", '

If you want to use these characters as part of the mentioned fields, replace:

- " with `&quot;`
- ' with `&apos;`
- < with `<`

- > with `>`
- & with `&`

WSUser

Field	Type	Description
username	String	The user name for the user being updated. This must be an existing user name.
password	String	The password to update.
isInternal	Boolean	An attribute describing whether the user is internal. `false` means that the user is external.
isEnabled	Boolean	An attribute describing whether the user will be enabled.
groups	String[]	A list of the group names to which the user will belong. If you leave this input empty, the user will belong to the `EVERYBODY` group, as it is the default behavior in Central UI. The group names are not case sensitive.

Note that:

- Any field, except `username`, can be empty. The empty fields' values remain unchanged.
- The built-in `admin` user cannot be updated.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_USERS` capability.
- The user name is empty.
- The password, if provided, has less than 6 characters.
- The password is empty and `isInternal` is `true`.
- The password is not empty and `isInternal` is `false`.
- The list of groups contains group names that do not exist.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="https://host:8443/PAS/services/WSCentralService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:updateUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <user xsi:type="wsc:WSUser">
```

```
            <username xsi:type="soapenc:string">user</username>
            <password xsi:type="soapenc:string">new_password</password>
            <isInternal xsi:type="soapenc:boolean"></isInternal>
            <isEnabled xsi:type="soapenc:boolean"></isEnabled>
            <groups xsi:type="wsc:ArrayOf_xsd_string" soapenc:arrayType="xsd:string[]"/>
         </user>
      </wsc:editUser>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:updateUserResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

deleteUser

This method deletes a user account:

```
Boolean deleteUser(String username)
```

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_USERS` capabilities.

This method uses the SOAP protocol, sending data in an XML format. Therefore, some special characters are not supported in the `username` string field.

The character set not being supported includes:

- the non-printable characters.
- `&`, `>`, `<`, `"`, `'`

If you want to use these characters as part of the mentioned fields, replace:

- " with `&quot;`
- ' with `&apos;`
- < with `<`
- > with `>`
- & with `&`

Inputs

Input	Type	Description
username	String	The user name for the user being deleted.

Outputs

Output	Type	Description
Result	Boolean	`true`, if the deletion was successful. `false`, if the user was not deleted. This can happen if the provided user name does not refer to an existing user.

Note that:

- The built-in `admin` user cannot be deleted.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_USERS` capability.
- The user name is empty.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="https://host:8443/PAS/services/WSCentralService">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:deleteUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <username xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">user</username>
      </wsc:deleteUser>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:deleteUserResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <deleteUserReturn xsi:type="xsd:boolean">true</deleteUserReturn>
      </ns1:deleteUserResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getUserGroups

This method gets the groups of a specific user:

```
String[ ] getUserGroups(String username)
```

When editing a user, the information regarding that user is overwritten with the one provided in the request. This method is useful if you needed to modify the `groups` attribute of a certain user.

For example, if you need to add a group to one user, use this method to retrieve the groups, add the new one and, in the request for updating a user, put the result in the `groups` field.

This method allows to retrieve all groups of a user, including `EVERYBODY`.

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `MANAGE_USERS` capabilities.

Inputs

Input	Type	Description
username	String	The user name for the user who's groups are being retrieved.

Outputs

Output	Type	Description
Result	String[]	An array of String elements, representing the group names, in upper case.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails: the authenticated user does not have `MANAGE_USERS` capability.
- The user name is empty.
- The user name does not exist.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getUserGroups soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <username xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">user_test</username>
      </wsc:getUserGroups>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getUserGroupsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getUserGroupsReturn soapenc:arrayType="soapenc:string[4]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
```

```
            <getUserGroupsReturn xsi:type="soapenc:string">LEVEL_THREE</getUserGroupsReturn>
            <getUserGroupsReturn xsi:type="soapenc:string">AUDITOR</getUserGroupsReturn>
            <getUserGroupsReturn xsi:type="soapenc:string">GR1</getUserGroupsReturn>
            <getUserGroupsReturn xsi:type="soapenc:string">LEVEL_ONE</getUserGroupsReturn>
         </getUserGroupsReturn>
      </ns1:getUserGroupsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Repositories

The methods documented in this section enable you to handle folders:

createFolder

This method creates a folder within a repository:

```
Boolean createFolder(String)
```

Inputs

Input	Type	Description
path	String	The full path of the new folder. If sections of the path are missing, they are created. If the folder exists, nothing happens. Folders can be created only in the **Library** section. The specified path needs to start with `/Library` or `Library`.

Outputs

Output	Type	Description
Result	Boolean	The folder was created (`true`) or the folder already exists (`false`).

Exceptions

Exceptions are thrown if:

- The path is invalid.

  A path is invalid if it is null, blank, contains illegal characters, has a section which is a reserved word or has a section which is longer than 255 characters.

The legal characters are alphanumeric, spaces or the following: `_, __, -, ¯, :, :, ., ·, ., ., ◦, ▪, ·, (, ), (, ), [, ], [, ], {, }, {, }`

The reserved words are:

- .attic
- .section
- .section.xml
- .xml

- The user is not allowed to create the folder.
- The path contains a sealed folder.

moveFolder

This method moves a folder within a repository:

```
String moveFolder(String sourcePath, String destinationPath, Boolean overwrite)
```

Inputs

Input	Type	Description
sourcePath	String	The source path of the folder to move.
destinationPath	String	The location where to move the folder.
overwrite	Boolean	Specifies whether to continue if there is a folder, at the `destinationPath` location, having the same name as the folder you are moving.

Outputs

Output	Type	Description
Result	String	The path to the moved folder.

Exceptions

`RemoteException` is thrown if:

- `sourcePath` does not refer to a folder.
- `destinationPath` contains a folder having the same name as the source folder and `overwrite` is set to `false`.
- The destination contains an object, other than a folder, having the same name as the source folder.
- The user does not have appropriate permissions.

search

This method retrieves information about the objects in the **Library** repository folder:

```
WSFlow[] search(String queryString, String basePath)
```

```
WSFlow {
       String flowName
       String flowPath
       String flowUUID
}
```

You can create queries using the following criteria:

- Category
- Description
- UUID
- Inputs
- Name
- Domain

Examples of queries searching in the root repository folder:

1. Search by unique UUID:
   `search("UUID:f5dbab66-29ba-461a-8a9c-e720226915b9", null)`
2. Search by name:
   `search("NAME:\"Attach Software Policy\"", null)`
3. Search by category:
   `search("CATEGORY:\"Windows OS\"", null)`

Inputs

Input	Type	Description
queryString	String	The string used to query the repository. This query is based on the `Lucene` index search. For additional information, see: http://lucene.apache.org/java/docs/queryparsersyntax.html
basePath	String	The path to the folder where the search is performed. The path must be under **Library**. If left empty, the search takes place in **Library**.

Outputs

Output	Type	Description
Result	WSFlow[]	An array containing the `WSFlow` objects that match the query.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:search soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <queryString xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">NAME:\windows*</queryString>
         <basePath xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"></basePath>
      </wsc:search>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:searchResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <searchReturn soapenc:arrayType="ns2:WSFlow[32]" xsi:type="soapenc:Array"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">0460fa6e-f3cc-4eb2-a709-4f449346b4fa</UUID>
               <name xsi:type="soapenc:string">Windows Diagnostic</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Deprecated</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">fbddc27f-666b-44f3-a1c6-5da924508d6b</UUID>
               <name xsi:type="soapenc:string">Windows Diagnostic</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Diagnostics</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">19c48cf1-3b31-4b6c-81d7-0a99f02a92c7</UUID>
               <name xsi:type="soapenc:string">Clean Windows Files</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Deprecated</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">fdc0c5b4-37dc-4c79-a155-ea86d9b9c75f</UUID>
               <name xsi:type="soapenc:string">Windows Cluster Reporting</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Cluster</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">9dd191c7-4a50-4e08-8d92-13d8ceb0c387</UUID>
               <name xsi:type="soapenc:string">Windows RAS Traceroute Sample</name>
               <path
xsi:type="soapenc:string">/Library/Operations/Network/Samples/Deprecated</path>
```

```
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">a1ad2af5-b2c0-4845-a38a-b6eeef15f2c1</UUID>
               <name xsi:type="soapenc:string">Bundle Windows Instance Sample</name>
               <path
xsi:type="soapenc:string">/Library/Integrations/Amazon/EC2/Windows/Samples</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">f53ed4a3-07a9-43ce-8452-932769ab919e</UUID>
               <name xsi:type="soapenc:string">Windows Health Check</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Deprecated</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">1883432b-d89f-4438-971a-8d136eeed31b</UUID>
               <name xsi:type="soapenc:string">Windows Event Log Diagnostic</name>
               <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Event Logs</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">23848cb8-cf55-46f0-b9f0-2d88313ac4d9</UUID>
               <name xsi:type="soapenc:string">Reset Windows Service Credentials</name>
               <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Services</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">2da44b9a-2161-47c5-94a9-864920bedef8</UUID>
               <name xsi:type="soapenc:string">Check Windows CPU</name>
               <path xsi:type="soapenc:string">/Library/Templates</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">77a0d53c-c9c0-4f72-922f-d121659d595b</UUID>
               <name xsi:type="soapenc:string">Check for Windows Event</name>
               <path xsi:type="soapenc:string">/Library/Templates</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">a88033fb-4d7a-4f3b-8a57-c65e569824c7</UUID>
               <name xsi:type="soapenc:string">Remove from Windows Cluster</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/IIS/Utility</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">e3bf9bf7-e630-4303-8a8a-73eff7a7895d</UUID>
               <name xsi:type="soapenc:string">Windows Delete or Zip</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Utility</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">084ebdbd-912a-4d5f-bcc9-ce75c178ed63</UUID>
               <name xsi:type="soapenc:string">Windows Health Check (1)</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Deprecated</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">3ed79da2-5f42-49b8-945b-4d2bde82274d</UUID>
               <name xsi:type="soapenc:string">Restart Windows Service</name>
               <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Deprecated</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">aa4e0884-f11c-4511-8af2-d9497910257d</UUID>
               <name xsi:type="soapenc:string">Get Windows Processes</name>
```

```
            <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Processes</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">e09ef538-4885-47c5-aaa0-6c7250b4187f</UUID>
            <name xsi:type="soapenc:string">Clean Windows Files</name>
            <path xsi:type="soapenc:string">/Library/ITIL/Incident
Management/Servers</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">1c7b5892-aada-464b-b52a-ddfc78cf0c92</UUID>
            <name xsi:type="soapenc:string">Restart Windows Server</name>
            <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/State change</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">43e8b556-3e7b-4411-b1f6-7bb373c60183</UUID>
            <name xsi:type="soapenc:string">Windows Delete or Zip</name>
            <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Deprecated</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">fb438a7c-ad48-4a86-b220-906b18b6acab</UUID>
            <name xsi:type="soapenc:string">Clean Windows Files</name>
            <path xsi:type="soapenc:string">/Library/ITIL/Incident
Management/Servers/Deprecated</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">3541d63f-603a-449b-9d43-8e57d7d61482</UUID>
            <name xsi:type="soapenc:string">Windows Health Check</name>
            <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">47345b0c-a7b3-4d11-ae60-95d83b7b4f81</UUID>
            <name xsi:type="soapenc:string">Windows Cluster Diagnostic</name>
            <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Diagnostics</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">9d5e3acc-0d42-49dd-b6bc-bf57dacdffa2</UUID>
            <name xsi:type="soapenc:string">Windows Cluster Consistency Check</name>
            <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Cluster</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">abbee20f-7995-47c5-910a-cd54a3691a01</UUID>
            <name xsi:type="soapenc:string">Windows Delete or Zip</name>
            <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Samples</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">2340d930-298a-4ac6-a830-c272693ce42e</UUID>
            <name xsi:type="soapenc:string">Windows RAS Ping Sample</name>
            <path
xsi:type="soapenc:string">/Library/Operations/Network/Samples/Deprecated</path>
         </searchReturn>
         <searchReturn xsi:type="ns2:WSFlow">
            <UUID xsi:type="soapenc:string">2c813648-c4ea-44fb-855f-988bb0b535ea</UUID>
            <name xsi:type="soapenc:string">Detect Restarted Windows Server</name>
            <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
```

```
Management/Event Logs</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">1d8b1630-eed6-4094-b3de-7de6a31e661f</UUID>
               <name xsi:type="soapenc:string">HP OVO Windows Notify</name>
               <path xsi:type="soapenc:string">/Library/Integrations/Hewlett-
Packard/Operations Manager/Deprecated/Windows Only/Deprecated</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">70869f21-9fc8-4973-9fd1-9c3fbdeaf550</UUID>
               <name xsi:type="soapenc:string">Restart Windows Service</name>
               <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Services</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">fe4a1f40-76e0-48ef-8ac3-055637f6ad38</UUID>
               <name xsi:type="soapenc:string">Clean Windows Files</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/Utility</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">f615d305-e32c-43d1-ae81-1de47a722e37</UUID>
               <name xsi:type="soapenc:string">Restart Windows Server with Wait</name>
               <path xsi:type="soapenc:string">/Library/Accelerator Packs/Operating
Systems/Windows/State change</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">5a9d5022-024f-447b-9e77-ff81f1dae9ed</UUID>
               <name xsi:type="soapenc:string">Reboot Windows Server with MOM</name>
               <path xsi:type="soapenc:string">/Library/Integrations/Microsoft/Microsoft
Operations Manager/Samples</path>
            </searchReturn>
            <searchReturn xsi:type="ns2:WSFlow">
               <UUID xsi:type="soapenc:string">56fac8a8-c84f-4589-8789-308b42b67b2a</UUID>
               <name xsi:type="soapenc:string">Iterative Windows Event Log Diagnostic</name>
               <path xsi:type="soapenc:string">/Library/Operations/Operating Systems/Windows
Management/Event Logs</path>
            </searchReturn>
         </searchReturn>
      </ns1:searchResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

list

This method retrieves the content of a folder from the repository:

```
WSFolderContent list(String basePath)
```

```
WSFolderContent {
       WSFolderDetails[] folders
       WSFlow[] flows
       String folderName
       boolean folderFound
}
WSFolderDetails {
       String folderName
       String folderPath
       String lastModifiedBy
       long lastModifiedTime - milliseconds from 1/1/1970
```

```
}
WSFlow {
       String flowName;

       String flowPath;
       String flowUUID;
}
```

Inputs

Input	Type	Description
basePath	String	The path to the folder whose content is to be listed. The path must be under the **Library**.

Outputs

Output	Type	Description
Result	WSFolderContent	The `WSFolderContent` object containing the list of subfolders, the list of repository objects (if the folder was found) and the folder name.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:list soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <basePath xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/Tutorials</basePath>
      </wsc:list>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:listResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <listReturn xsi:type="ns2:WSFolderContent"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <flows soapenc:arrayType="ns2:WSFlow[1]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
```

```
               <flows xsi:type="ns2:WSFlow">
                  <UUID xsi:type="soapenc:string">31ea143f-2ae5-41bf-b44b-7a8c423011c5</UUID>
                  <name xsi:type="soapenc:string">Restart Service - Tutorial Flow</name>
                  <path xsi:type="soapenc:string">/Library/Tutorials</path>
               </flows>
            </flows>
            <folderFound xsi:type="xsd:boolean">true</folderFound>
            <folderName xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/Tutorials</folderName>
            <folders soapenc:arrayType="ns2:WSFolderDetails[1]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
               <folders xsi:type="ns2:WSFolderDetails">
                  <lastModifiedBy xsi:type="soapenc:string">admin</lastModifiedBy>
                  <lastModifiedTime xsi:type="xsd:long">1340988262933</lastModifiedTime>
                  <name xsi:type="soapenc:string">Deprecated</name>
                  <path xsi:type="soapenc:string">/Library/Tutorials/Deprecated</path>
               </folders>
            </folders>
         </listReturn>
      </ns1:listResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getCategories

This method retrieves the categories existing in the Central **Library**:

```
String [] getCategories()
```

Outputs

Output	Type	Description
Result	String []	An array containing the names of the categories found in the repository root folder.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getCategories soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getCategoriesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getCategoriesReturn soapenc:arrayType="soapenc:string[11]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getCategoriesReturn xsi:type="soapenc:string">Application Server (.NET)
</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Application Server (J2EE)
</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Clusters</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Database
Server</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Linux OS</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Messaging</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Network</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Security</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Unix OS</getCategoriesReturn>
            <getCategoriesReturn
xsi:type="soapenc:string">Virtualization</getCategoriesReturn>
            <getCategoriesReturn xsi:type="soapenc:string">Windows OS</getCategoriesReturn>
         </getCategoriesReturn>
      </ns1:getCategoriesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getAttributes

This method retrieves the attributes of a flow, operation or folder.

```
WSAttribute[] getAttributes(String identifier, String[] attributes)
```

```
WSAttribute {
                    String name,  // name of the attribute
                    String value   // value of the attribute
}
```

The attributes include:

- uuid
- path
- type
- description
- hidden (if the object is set to be hidden)
- version
- changedby
- lastmodified
- permissions

The `uuid` and `path` attributes enable you to retrieve the object's path based on the `uuid` and vice-versa.

Inputs

Input	Type	Description
iden-tifier	String	The object path or UUID.
attrib-utes	String[]	An array of Strings, containing the attributes to retrieve. The available values (case insensitive) are: • type • description • hidden • version • changedby • lastmodified • permissions

Outputs

Output	Type	Description
Result	WSAttribute[]	An array of pairs (attribute name and value) for each requested attribute.

Attribute values:

- `uuid` - The object UUID.
- `path` - The path to the object in the repository.
- `type` - The type: `Flow`, `Operation` or `Folder`.
- `hidden` - `true`, if the object is hidden or `false`, otherwise.
- `description` - The description of the object.
- `changedby` - The user name of the last user to change this object.
- `lastmodified` - The timestamp of the latest change,in milliseconds, since January 1, 1970 UTC.
- `permissions` - A list of permissions for this object, formatted as `group1 permissions1, group2 permissions2`.

  `permissionsX` - A four characters String, indicating the permissions in the order: `read, write, execute, link. 'r' – read, 'w' – write, 'x' – execute, 'l' – link, '-' – no permission`

  Example:

  `"EVERYBODY" "r-x-" "ADMINISTRATOR" "rwxl"`

- The users from the `EVERYBODY` group can read and execute the object.
- The users from the `ADMINISTRATOR` group can read, write, execute and link the object.

Exceptions

`AxisFault` is thrown if:

- The `identifier` is null.
- The list of attributes is invalid:
  - Null.
  - Empty list.
  - Contains an invalid attribute.
  - Contains an attribute multiple times.
- The object does not exist or it is not a flow, operation or folder.
- The user does not have read permissions for the object. The actual error is `Object not found`. The user does not see the object.

getPermissions

This method gets the permissions for objects such as:

- Flow.
- Folder.
- Operation.
- System Account.
- Remote Action Service.

Extra options:

- Retrieve only the groups on which object permissions are set, or retrieve all the groups, even those without any set permissions.
- Use Linux style permissions or normal permissions.

The method is:

```
String[] getPermissions(String identifier, boolean showAllGroups,
boolean useLinuxStyle)
```

Inputs

Input	Type	Description
identifier	String	The path or the UUID of the flow, folder, operation, system account or remote action service.

showAllGroups	Boolean	A flag indicating whether to show all the groups (`true`), or show only the groups for which the object has permissions set in the response (`false`).
useLinuxStyle	Boolean	A flag indicating whether to use Linux style permissions in the response (`true`), for example, `RW-L`, or use normal permissions (`false`), for example, `RWL`.

Outputs

Output	Type	Description
Result	WSGroupPermissions []	A `WSGroupPermissions` array, containing the associations between the group name and the group permissions.

Note that:

- The elements in the response are ordered alphabetically, containing elements from both groups: with permissions set and with permissions unset.
- The `getPermissions` response's first groups are the ones having permissions set.
- The `getPermissions` response's last groups are the ones having permissions unset.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails:
  - None of the groups containing the authenticated user has `AUTHOR` capability.
  - The user is not `PROMOTER` with `HEADLESS_FLOWS` capability.
  - The user does not have `R` permission for all the object ancestors.
  - The user does not have `R` permissions for the object.
- Object is not found neither by path, nor by UUID.
- The SOAP request is malformed:
  - Missing parameters.
  - Invalid parameter order.
  - Malformed parameters.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getPermissions soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <identifier xsi:type="soapenc:string"
```

```
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/Accelerator
Packs</identifier>
         <showAllGroups xsi:type="xsd:boolean">1</showAllGroups>
         <useLinuxStyle xsi:type="xsd:boolean">1</useLinuxStyle>
      </wsc:getPermissions>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getPermissionsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getPermissionsReturn soapenc:arrayType="xsd:anyType[10]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getPermissionsReturn xsi:type="ns2:WSGroupPermissions"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">EVERYBODY</groupName>
               <permissions xsi:type="xsd:string">rwxl</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns5:WSGroupPermissions"
xmlns:ns5="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">AUTHOR</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns6:WSGroupPermissions"
xmlns:ns6="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">AUTOR</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns7:WSGroupPermissions"
xmlns:ns7="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">DOMAIN GUESTS</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns8:WSGroupPermissions"
xmlns:ns8="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">EVALUATOR</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns10:WSGroupPermissions"
xmlns:ns10="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">LEVEL_ONE</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns11:WSGroupPermissions"
xmlns:ns11="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">LEVEL_THREE</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns12:WSGroupPermissions"
xmlns:ns12="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">LEVEL_TWO</groupName>
               <permissions xsi:type="xsd:string">----</permissions>
            </getPermissionsReturn>
            <getPermissionsReturn xsi:type="ns13:WSGroupPermissions"
xmlns:ns13="http://iconclude.com/webservices/rss/v2.0/soap">
               <groupName xsi:type="xsd:string">NOHEAD</groupName>
```

```
            <permissions xsi:type="xsd:string">----</permissions>
         </getPermissionsReturn>
         <getPermissionsReturn xsi:type="ns14:WSGroupPermissions"
xmlns:ns14="http://iconclude.com/webservices/rss/v2.0/soap">
            <groupName xsi:type="xsd:string">NOHEADLESS</groupName>
            <permissions xsi:type="xsd:string">----</permissions>
         </getPermissionsReturn>
      </getPermissionsReturn>
   </ns1:getPermissionsResponse>
</soapenv:Body>
</soapenv:Envelope>
```

setPermissions

This method sets the permissions with the given `groupPermissions`, for objects such as:

- Flow.
- Folder.
- Operation.
- System Account.
- Remote Action Service.

Extra options:

- Apply the permission changes to folder contents or referenced objects.
- Copy the permissions to the folder's children.

The method is:

```
setPermissions(String identifier, WSGroupPermissions[]
groupPermissions, boolean applyToAllContents, boolean
copyToNewContent)
```

```
WSGroupPermissions {
       String groupName
       String permissions
}
```

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `AUTHOR` capabilities.

WSGroupPermissions

Field	Type	Description
identifier	String	The path or the UUID of the flow, folder, operation, system account or remote action service.

groupPermissions	WSGroupPermissions []	An unordered `WSGroupPermissions` list, where: • `groupName` is a unique group name. • `permissions` is an unordered list of individual rights. For example, `EVERYBODY:RWXL, Group1:N` (N-none).
applyToAllContents	Boolean	This is effective when the identifier points to a folder, flow or operation. Otherwise, this input is ignored. If `true`, the following actions occur: • For folders, applies the given `groupPermissions` change and propagates it to child objects recursively. `Read-only` children are not modified. • For flows and operations, applies the given `groupPermissions` change and propagates it to referenced items.
copyToNewContent	Boolean	This is effective when the identifier points to a folder. If `true`, it applies to newly added objects inside the folder, giving them the folder's permissions.

Note that:

- Only changes are propagated, similar to Studio's options:
  - **Apply to All Contents** > **Changes**
  - **Apply to all referenced objects** > **Changes**
- Group names and the group permissions are not case sensitive.
- Permissions can be set in any order, for example: `RWXL` or `RXLW`.
- The `N` permission (none) cannot be used along with any other permission.
- The user can set permissions on all the groups displayed in the Studio UI, including the `EVERYBODY` group, if he:
  - Is a flow developer (has `AUTHOR` capabilities in any of his groups).
  - Has `RW` permissions on the object he is modifying.
  - Can view the object in Studio (the `R` right for object ancestors).
- As in Studio, the user cannot lock himself out (for all his groups, take `RW` permissions for the objects under modification).
- `Read-only` objects are not modified.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- Authorization fails:
  - None of the groups containing the authenticated user has `AUTHOR` capability.
  - The user is not `PROMOTER` with `HEADLESS_FLOWS` capability.
  - The user does not have `R` permission for all the object ancestors.
  - The user does not have `R` permissions for the object.
- The object is checked out.
- The object is sealed.
- The object is not found, neither by path, nor by UUID.
- The SOAP request is malformed:
  - No permission information given.
  - Duplicate groups.
  - Invalid object type. Valid object type values: flow, folder, operation, system account,remote action service
  - Invalid permission information.
  - Duplicate permission information.
  - Conflicting permission information.
  - Groups cannot be modified.
  - Groups do not exist.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:setPermissions soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <identifier xsi:type="soapenc:string">/Configuration/System Accounts</identifier>
         <groupPermissions xsi:type="wsc:ArrayOf_tns1_WSGroupPermissions"
soapenc:arrayType="cli:WSGroupPermissions[]"
xmlns:wsc="https://clusterlion:8443/PAS/services/WSCentralService"
xmlns:cli="http://client.wscentralservice.services.dharma.iconclude.com">
  <groupPermission>
   <groupName>LEVEL_ONE</groupName>
   <permissions>x</permissions>
  </groupPermission>
  <groupPermission>
   <groupName>LEVEL_TWO</groupName>
   <permissions>rw</permissions>
  </groupPermission>
  <groupPermission>
   <groupName>EVERYBODY</groupName>
   <permissions>RW</permissions>
```

```
  </groupPermission>
  </groupPermissions>
        <applyToAllContents xsi:type="xsd:boolean">1</applyToAllContents>
        <copyToNewContent xsi:type="xsd:boolean">1</copyToNewContent>
     </wsc:setPermissions>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:setPermissionsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

renameRepoEntity

This method renames an entity from the repository:

`renameRepoEntity(String identifier, String newName)`

Inputs

Input	Type	Description
identifier	String	The path or the UUID of the entity.
newName	String	The entity's new name.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- The user does not have `HEADLESS_FLOWS` and `AUTHOR` capabilities.
- The user does not have `R` and `W` permissions on the entity or `R` permissions on one of its ancestors.
- The entity is not found.
- The entity cannot be renamed because is checked out by another user.
- The entity is located in a sealed folder.
- The entity is the root **Library** section or is a **Configuration** section.
- Another entity, having the `newName` name, already exists in the entity folder.
- `newName` is the same as the current name.

- The new name is not a valid name. A valid name:
  - Has less than 255 characters.
  - Contains only alphanumeric, spaces or the following characters: '_', '__', '-', '—', ':', ':', '.', '·', '. ', '。', '·', '。', '▪', '•', '(', ')', '(', ')', '\[', '\]', ' [', ']', '{', '}', ' {', '}'

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:renameRepoEntity soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <identifier xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/123</identifier>
         <newName xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">AA</newName>
      </wsc:renameRepoEntity>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.generalException</faultcode>
         <faultstring>The object with identifier '/Library/123' was not found.</faultstring>
         <detail>
            <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">cduma</ns1:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

deleteRepoEntity

This method deletes an entity from the repository:

`deleteRepoEntity(String identifier)`

This method applies to the following entities:

- Flows.
- Operations.
- Folders.
- Configuration Elements:
  - Domain Terms.
  - Remote Action Services.
  - Scriptlets.

- Selection Lists.
- System Accounts.
- System Evaluators.
- System Filters.
- System Properties.

Inputs

Input	Type	Description
identifier	String	The path or the UUID of the entity.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- The user does not have `HEADLESS_FLOWS` and `AUTHOR` capabilities.
- The entity is not found.
- The user does not have `R` and `W` permissions on the entity and the entity's parent folder, or `R` permissions on its ancestors.
- In case of folders, the user cannot checkout any object in that folder. This means that the user does not have `R` and `W` permissions, or the folder is check out by another user.
- The entity cannot be deleted because it is checked out by another user.
- The entity is located in a sealed folder.
- The entity is the root **Library** section or is a **Configuration** section.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:deleteRepoEntity soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <identifier xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/abc</identifier>
      </wsc:deleteRepoEntity>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.generalException</faultcode>
```

```
         <faultstring>The object with identifier '/Library/abc' was not found.</faultstring>
         <detail>
            <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">cduma</ns1:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

updateDescription

This method updates the description of an entity from the OO repository

```
Boolean updateDescription(String identifier, String description)
```

Note: This method can be ran only by an authenticated user having `HEADLESS_FLOWS` and `AUTHOR` capabilities.

Only the entities that have a description can be updated. The method applies to the following entities:

- Flows.
- Operations.
- Folders.
- Configuration Elements:
  - Domain Terms.
  - Remote Action Services.
  - Scriptlets.
  - Selection Lists.
  - System Accounts.
  - System Evaluators.
  - System Filters.
  - System Properties.

Characters like '<', '>' should be encoded `(<,>)` in order to send them correctly via SOAP.

Inputs

Input	Type	Description
identifier	String	The path or the UUID of the entity.
description	String	The new description. It can be empty (to erase the description), but cannot be null. The maximum description length is of 16384 characters.

Outputs

Output	Type	Description

Result	Boolean	Returns `true` if the description was updated, and `false` if the entity was not found or is not visible to the user.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- The user does not have `HEADLESS_FLOWS` and `AUTHOR` capabilities.
- The user does not have `R` and `W` permissions on the entity.
- The object being updated does not have a description property.
- The object being updated is a group or a configuration section.
- The entity cannot be updated because it is checked out by another user.
- The entity is in a sealed library section.
- The description is longer than 16384 characters.
- The description contains invalid XML characters. For example, vertical tab.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:updateDescription
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <identifier xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">/Library/Tutorials</identifier>
         <description xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">This is a folder for
tutorials</description>
      </wsc:updateDescription>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:updateDescriptionResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <updateDescriptionReturn xsi:type="soapenc:boolean"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">true</updateDescriptionReturn>
      </ns1:updateDescriptionResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Runs

The classes and the methods documented in this section enable you to handle OO runs.

Classes

- "WSRunParameters" below.
- "WSRunParametersEx" on page 184.
- "WSRunHandle" on page 186.

Methods

- "pauseRun" on page 189.
- "resumeRun" on page 190.
- "cancelRun" on page 192.
- "runFlow" on page 193.
- "runFlowEx" on page 195.
- "getFlowRunHistory" on page 199.
- "getFlowRunHistoryByRunId" on page 202.
- "getFlowsRunHistory" on page 204.
- "getRunStatus" on page 208.
- "getRunStatusEx" on page 210.
- "getStatusForRuns" on page 214.

Classes for handling runs

WSRunParameters

```
com.iconclude.dharma.services.wscentralservice

Class WSRunParameters

java.lang.Object
        |-com.iconclude.dharma.services.wscentralservice.WSRunParameters

---------------------------------------------------------------------
```

Direct Known Subclasses

```
WSRunParametersEx

---------------------------------------------------------------------

public class WSRunParameters
 extends java.lang.Object
```

This object is a wrapper for the parameters needed by the service to run a flow. The client compiles this object as parameter to the **runFlow** API service call.

Constructor Summary

`WSRunParameters()`

Methods Summary

Output	Method
`WSFlowInput[]`	`getFlowInputs()`
`java.lang.String`	`getRunName()`
`java.lang.String`	`getUuid()`
`boolean`	`isAsync()`
`boolean`	`isStartPaused()`
`boolean`	`isTrackStatus()`
`void`	`setAsync(boolean async)`
`void`	`setFlowInputs(WSFlowInput[] flowInputs)`
`void`	`setRunName(java.lang.String runName)`
`void`	`setStartPaused(boolean startPaused)`
`void`	`setTrackStatus(boolean trackStatus)`
`void`	`setUuid(java.lang.String uuid)`

Constructor Detail

`WSRunParameters()`

Methods Detail

getUuid

`getUuid()`

Returns:

String, representing the flow's UUID.

---

setUuid

```
void setUuid(String uuid)
```

Parameters:

`uuid` - The UUID of the flow to run.

---

getRunName

```
String getRunName()
```

Returns:

String, representing the run name.

---

setRunName

```
void setRunName(String runName)
```

Parameters:

`runName` - The name of the run. If you do not set the run name, or if you set as an empty String, the run name is: `flowName_${runID}`

---

getFlowInputs

```
WSFlowInput[] getFlowInputs()
```

Returns:

A `WSFlowInput` array wrapping the flow parameters.

---

setFlowInputs

```
void setFlowInputs(WSFlowInput[] flowInputs)
```

Parameters:

`flowInputs` - An array containing the inputs/parameters of the flow to run.

---

isAsync

```
boolean isAsync()
```

Returns:

`true`, if the run was asynchronous and returned immediately.

`false`, if the run went to completion before returning.

---

setAsync

```
void setAsync(boolean async)
```

Parameters:

`async` - If `true`, run asynchronously and return immediately. Otherwise, run to completion before returning.

---

isTrackStatus

```
boolean isTrackStatus()
```

Returns:

`true`, if the live run was cached for status monitoring.

`false`, if the run status was retrieved from history.

---

setTrackStatus

```
void setTrackStatus(boolean trackStatus)
```

Parameters:

`trackStatus` - If `true`, the live run will be cached for status monitoring. Otherwise, the run status will be retrieved from history. If the client needs to track the status of its runs, it should set this parameter to `true` in order to reduce the database access and speed up the response time. If monitoring is not required, set to `false` to eliminate unnecessary memory allocations for caching.

---

isStartPaused

```
boolean isStartPaused()
```

Returns:

`true`, if the flow run started paused.

`false`, if the fow ran immediately.

---

setStartPaused

```
void setStartPaused(boolean startPaused)
```

Parameters:

`startPaused` - If `true`, the flow run will start paused. Otherwise, it will run immediately. This is ignored if the run is synchronous.

WSRunParametersEx

```
com.iconclude.dharma.services.wscentralservice
```

```
Class WSRunParametersEx
```

```
java.lang.Object
       |-com.iconclude.dharma.services.wscentralservice.WSRunParameters
       |-com.iconclude.dharma.services.wscentralservice.WSRunParametersEx
```

```
public class WSRunParametersEx
 extends WSRunParameters
```

This object is an extended wrapper for the parameters needed by the service to run a flow. The client compiles this object as a parameter to the **runFlowEx** API service call.

Constructor Summary

```
WSRunParametersEx()
```

Methods Summary

Type	Method
`boolean`	`isPrimaryResultWanted()`
`boolean`	`isRawResultWanted()`
`boolean`	`isStatusWanted()`
`void`	`setPrimaryResultWanted(boolean primaryResultWanted)`
`void`	`setRawResultWanted(boolean rawResultWanted)`
`void`	`setStatusWanted(boolean statusWanted)`

Methods inherited from `class` `com.iconclude.dharma.services.wscentralservice.WSRunParameters`:

- getFlowInputs
- getRunName
- getUuid
- isAsync
- isStartPaused
- isTrackStatus
- setAsync
- setFlowInputs
- setRunName
- setStartPaused
- setTrackStatus
- setUuid

Constructor Detail

WSRunParametersEx()

Methods Detail

isStatusWanted

```
boolean isStatusWanted()
```

Returns:

`true`, if the run status is requested.

`false`, otherwise.

---

setStatusWanted

```
void setStatusWanted(boolean statusWanted)
```

Parameters:

If `statusWanted` is `true`, the status of the run is displayed, including:

- the results
- the steps of the flow
- the report URL
- the run handle

- the status
- the status cursor

If `false`, only the run ID and the status cursor are returned.

---

isRawResultWanted

```
boolean isRawResultWanted()
```

Returns:

`true`, if the raw result of the run is to be returned in the response.

`false`, otherwise.

---

setRawResultWanted

```
void setRawResultWanted(boolean rawResultWanted)
```

Parameters:

If `rawResultWanted` is `true`, the raw result is returned in the response. If `false`, it is not returned.

---

isPrimaryResultWanted

```
boolean isPrimaryResultWanted()
```

Returns:

`true`, if the primary result of the run is to be returned in the response.

`false`, otherwise.

---

setPrimaryResultWanted

```
void setPrimaryResultWanted(boolean primaryResultWanted)
```

Parameters:

If `primaryResultWanted` is `true`, the primary result is returned in the response.If `false`, it is not returned.

WSRunHandle

`com.iconclude.dharma.services.wscentralservice`

```
Class WSRunHandle

java.lang.Object
        |-com.iconclude.dharma.services.wscentralservice.WSRunParameters
```

Direct Known Subclasses:

WSRunHandleEx

`public class` WSRunHandle
extends `java.lang.Object`

`WSRunHandle` is a wrapper object for the run ID and `WSRunStatusCursor`. It is used mainly to retrieve the run status and pause, cancel, and resume runs.

See also:

resumeRun, pauseRun, cancelRun, getRunStatus

Constructor Summary

```
WSRunHandle()

WSRunHandle(java.lang.String runID, WSRunStatusCursor statusCursor)

WSRunHandle(java.lang.String runID, WSRunStatusCursor statusCursor,
java.lang.String runName)
```

Methods Summary

Type	Method
java.lang.String	getRunID()
java.lang.String	getRunName()
WSRunStatusCursor	getStatusCursor()
void	setRunID(java.lang.String runID)
void	setRunName(java.lang.String runName)
void	setStatusCursor(WSRunStatusCursor statusCursor)

Constructor Detail

```
WSRunHandle()
```

`WSRunHandle(String runID,WSRunStatusCursor statusCursor)`

Parameters:

`runID` - String, representing Central's unique run ID, UUID.

`statusCursor` - The `WSRunStatusCursor`, representing the current (or desired) step index in the history of the run.

`WSRunHandle (String runID, WSRunStatusCursor statusCursor, String runName)`

Parameters:

`runID` - String, representing Central's unique run ID, UUID.

`statusCursor` - The `WSRunStatusCursor`, representing current (or desired) step index in the history of the run.

`runName` - String, representing Central's unique run name.

Methods Detail

getRunID

```
String getRunID()
```

Returns:

String, representing the run ID.

---

setRunID

```
void setRunID(String runID)
```

Parameters:

`runID` - String, representing the run ID.

---

getStatusCursor

```
WSRunStatusCursor getStatusCursor()
```

Returns:

`WSRunStatusCursor`, representing the current step index in the history of the run.

---

setStatusCursor

```
void setStatusCursor(WSRunStatusCursor statusCursor)
```

Parameters:

`statusCursor`, representing the desired step index in the history of the run.

---

getRunName

```
String getRunName()
```

Returns:

String, representing the run name.

---

setRunName

```
void setRunName(String runName)
```

Set the run name.

Parameters:

`runName` - String, representing Central's unique run name.

Methods for handling runs

pauseRun

This method pauses a run:

```
WSRunCtrlResult pauseRun(WSRunCtrlParams runCtrlParams)
```

```
WSRunCtrlResult {
       boolean success
       String errorMsg
}
WSRunCtrlParams {
       boolean synchronus
       String runID
}
```

Inputs

Input	Type	Description
runCtrlParams	WSRunCtrlParams	Details about the run you want to pause: the run ID and whether to wait for the operation to finish before returning.

Outputs

Output	Type	Description
Result	WSRunCtrlResult	The result of the operation and, if the operation failed, the error message.

Exceptions

`AxisFault` is thrown if:

- `runCtrlParams` is null.
- The provided run ID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:pauseRun soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runCtrlParams xsi:type="soap:WSRunCtrlParams"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">1466</runID>
            <sync xsi:type="xsd:boolean">true</sync>
         </runCtrlParams>
      </wsc:pauseRun>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:pauseRunResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <pauseRunReturn xsi:type="ns2:WSRunCtrlResult"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <errorMsg xsi:type="soapenc:string" xsi:nil="true"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
            <success xsi:type="xsd:boolean">true</success>
         </pauseRunReturn>
      </ns1:pauseRunResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

resumeRun

This method resumes a run:

```
WSRunCtrlResult resumeRun(WSRunCtrlParams runCtrlParams)
```

```
WSRunCtrlResult {
       boolean success
       String errorMsg
}
WSRunCtrlParams {
       boolean synchronus
       String runID
}
```

Inputs

Input	Type	Description
runCtrlParams	WSRunCtrlParams	Details about the run you want to resume: the run ID and whether to wait for the operation to finish before returning.

Outputs

Output	Type	Description
Result	WSRunCtrlResult	The result of the operation and, if the operation failed, the error message.

Exceptions

`AxisFault` is thrown if:

- `runCtrlParams` is null.
- The provided run ID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:resumeRun soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runCtrlParams xsi:type="soap:WSRunCtrlParams"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">2</runID>
            <sync xsi:type="xsd:boolean">false</sync>
         </runCtrlParams>
      </wsc:resumeRun>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
```

```
      <ns1:resumeRunResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <resumeRunReturn xsi:type="ns2:WSRunCtrlResult"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <errorMsg xsi:type="soapenc:string" xsi:nil="true"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
            <success xsi:type="xsd:boolean">true</success>
         </resumeRunReturn>
      </ns1:resumeRunResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

cancelRun

This methods enables you to cancel a run:

```
WSRunCtrlResult cancelRun(WSRunCtrlParams runCtrlParams)
```

```
WSRunCtrlResult {
       boolean success
       String errorMsg
}
WSRunCtrlParams {
       boolean synchronus
       String runID
}
```

Inputs

Input	Type	Description
runCtrlParams	WSRunCtrlParams	Details about the run you want to cancel: the run ID and whether to wait for the operation to finish before returning.

Outputs

Output	Type	Description
Result	WSRunCtrlResult	The result of the operation and, if the operation failed, the error message.

Exceptions

`AxisFault` is thrown if:

- `runCtrlParams` is null.
- The provided run ID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:cancelRun soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runCtrlParams xsi:type="soap:WSRunCtrlParams"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">1466</runID>
            <sync xsi:type="xsd:boolean">true</sync>
         </runCtrlParams>
      </wsc:cancelRun>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:cancelRunResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <cancelRunReturn xsi:type="ns2:WSRunCtrlResult"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <errorMsg xsi:type="soapenc:string" xsi:nil="true"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
            <success xsi:type="xsd:boolean">true</success>
         </cancelRunReturn>
      </ns1:cancelRunResponse>
   </soapenv:Body>
```

runFlow

This method runs a flow:

```
WSRunHandle runFlow(WSRunParameters params)
```

```
WSRunHandle {
       String runID
       WSRunStatusCursor statusCursor

}
WSRunStatusCursor {
       int cursorPosition - the step index in the run history. The step count starts from 0.
}
WSRunParameters {
       String Uuid
       String runName;
       WSFlowInput[] flowInputs
       boolean async
       boolean trackStatus
       Boolean startPaused
}
WSFlowInput {
       String name
       String value
       boolean encrypted
       String[] selectedValues
}
```

Inputs

Input	Type	Description
params	WSRunParameters	Details about the flow you want to run: • The flow UUID. • The run name that will be given to the flow. • The flow inputs. • Whether the flow will run synchronous or asynchronous. • If the live run will be cached for status monitoring, or will it be retrieved from history. • Whether the flow will start in the paused state.

Outputs

Output	Type	Description
Result	WSRunHandle	The flow run ID and the index from the run history.

Exceptions

`AxisFault` is thrown if:

- `Params` is null.
- The provided UUID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:runFlow soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <params xsi:type="soap:WSRunParameters"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <async xsi:type="xsd:boolean">true</async>
            <flowInputs xsi:type="wsc:ArrayOf_tns2_WSFlowInput"
soapenc:arrayType="soap:WSFlowInput[]" xmlns:wsc="https://my-
ooserver.myco.com:8443/PAS/services/WSCentralService">
               <input>
                       <name>host</name>
                       <value>localhost</value>
               </input>
           </flowInputs>
            <runName xsi:type="xsd:string">new_name</runName>
            <startPaused xsi:type="xsd:boolean">false</startPaused>
```

```
            <trackStatus xsi:type="xsd:boolean">true</trackStatus>
            <uuid xsi:type="xsd:string">3541d63f-603a-449b-9d43-8e57d7d61482</uuid>
         </params>
      </wsc:runFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:runFlowResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <runFlowReturn xsi:type="ns2:WSRunHandle"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">920</runID>
            <statusCursor xsi:type="ns2:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">0</cursorPosition>
            </statusCursor>
         </runFlowReturn>
      </ns1:runFlowResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

runFlowEx

This method starts the run of a flow, along with specifying the returned result:

```
WSRunHandleEx  runFlowEx(WSRunParametersEx paramsEx)
```

```
WSRunHandleEx {
       WSRunStatusEx runStatusEx
       WSStatusOptionsEx statusOptions
}
WSRunStatusEx {
       String primaryResult
       String rawResult
       WSStatusOptionsEx statusOptions
}
WSStatusOptionsEx {
       boolean rawResultWanted
       boolean primaryResultWanted
}
WSRunParametersEx {
       String Uuid
       String runName;
       WSFlowInput[] flowInputs
       boolean async
       boolean trackStatus
       Boolean startPaused
       boolean statusWanted
       boolean rawResultWanted
       boolean primaryResultWanted

}
WSFlowInput {
       String name
       String value
```

```
        boolean encrypted
        String[] selectedValues
}
```

Inputs

Input	Type	Description
params	WSRunParametersEx	Details about the flow you want to run: • The flow UUID. • The run name that will be given to the flow. • The flow inputs. • Whether the flow will run synchronous or asynchronous. • If the live run will be cached for status monitoring, or will it be retrieved from history. • Whether the flow will start in the paused state. • Whether the method will return the raw result. • Whether the method will return the status of the run. • Wheter the method will return the primary result.

Outputs

Output	Type	Description
Result	WSRunHandleEx	The flow run ID and the index from the run history.

Exceptions

`AxisFault` is thrown if:

- `Params` is null.
- The provided UUID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <wsc:runFlowEx soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <paramsEx xsi:type="soap:WSRunParametersEx"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <async xsi:type="xsd:boolean">true</async>
            <flowInputs xsi:type="wsc:ArrayOf_tns2_WSFlowInput"
soapenc:arrayType="soap:WSFlowInput[]" xmlns:wsc="https://my-
ooserver.myco.com:8443/PAS/services/WSCentralService">
               <input>
                       <name>host</name>
                       <value>localhost</value>
                       <encrypted>false</encrypted>
               </input>
           </flowInputs>
            <runName xsi:type="xsd:string">new run</runName>
            <startPaused xsi:type="xsd:boolean">false</startPaused>
            <trackStatus xsi:type="xsd:boolean">true</trackStatus>
            <uuid xsi:type="xsd:string">3541d63f-603a-449b-9d43-8e57d7d61482</uuid>
            <primaryResultWanted xsi:type="xsd:boolean">true</primaryResultWanted>
            <rawResultWanted xsi:type="xsd:boolean">false</rawResultWanted>
            <statusWanted xsi:type="xsd:boolean">true</statusWanted>
         </paramsEx>
      </wsc:runFlowEx>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:runFlowExResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <runFlowExReturn xsi:type="ns2:WSRunHandleEx"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">31075</runID>
            <runStatusEx xsi:type="ns2:WSRunStatusEx">
               <primaryResult xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
               <rawResult xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
               <runHandle xsi:type="ns2:WSRunHandle">
                  <runID xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">31075</runID>
                  <statusCursor xsi:type="ns2:WSRunStatusCursor">
                     <cursorPosition xsi:type="xsd:int">10</cursorPosition>
                  </statusCursor>
               </runHandle>
               <runReportUrl xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">https://my-
ooserver.myco.com:8443/PAS/app?service=RCLinkService/ReportLinkDispatch&amp;sp=SINDIVIDUAL_
REPAIR_LEVEL&amp;sp=S3541d63f-603a-449b-9d43-
8e57d7d61482&amp;sp=l0&amp;sp=l43304&amp;sp=l31075</runReportUrl>
               <runResponse xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
               <runResumeUrl xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">https://my-
ooserver.myco.com:8443/PAS-
/app?service=RCLinkService/FlowLinkDispatch&amp;sp=SRESUME&amp;sp=l31075</runResumeUrl>
               <status xsi:type="xsd:int">0</status>
               <statusOptions xsi:type="ns2:WSStatusOptionsEx">
```

```
                  <primaryResultWanted xsi:type="xsd:boolean">true</primaryResultWanted>
                  <rawResultWanted xsi:type="xsd:boolean">false</rawResultWanted>
               </statusOptions>
               <steps soapenc:arrayType="ns2:WSRunStepDetails[10]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078104000</endTime>
                     <flowName xsi:type="soapenc:string">Windows Health Check</flowName>
                     <name xsi:type="soapenc:string">Windows Health Check</name>
                     <runStepLevel xsi:type="xsd:int">0</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078104000</startTime>
                     <stepResponse xsi:type="soapenc:string"/>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078107000</endTime>
                     <flowName xsi:type="soapenc:string">Windows Health Check</flowName>
                     <name xsi:type="soapenc:string">Ping Target System</name>
                     <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078104000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078108000</endTime>
                     <flowName xsi:type="soapenc:string">Windows Health Check</flowName>
                     <name xsi:type="soapenc:string">Get System Uptime</name>
                     <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078107000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078107000</endTime>
                     <flowName xsi:type="soapenc:string">Formatted WMI Query (1)</flowName>
                     <name xsi:type="soapenc:string">WMI Query</name>
                     <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078107000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078107000</endTime>
                     <flowName xsi:type="soapenc:string">Formatted WMI Query (1)</flowName>
                     <name xsi:type="soapenc:string">WMIQueryResultsTransformer</name>
                     <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078107000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078108000</endTime>
                     <flowName xsi:type="soapenc:string">Formatted WMI Query (1)</flowName>
                     <name xsi:type="soapenc:string">Resolved : success</name>
```

```
                     <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078108000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078108000</endTime>
                     <flowName xsi:type="soapenc:string">Windows Health Check</flowName>
                     <name xsi:type="soapenc:string">Get Host Name</name>
                     <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078108000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078108000</endTime>
                     <flowName xsi:type="soapenc:string">Formatted WMI Query (1)</flowName>
                     <name xsi:type="soapenc:string">WMI Query</name>
                     <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078108000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078108000</endTime>
                     <flowName xsi:type="soapenc:string">Formatted WMI Query (1)</flowName>
                     <name xsi:type="soapenc:string">WMIQueryResultsTransformer</name>
                     <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078108000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349078108000</endTime>
                     <flowName xsi:type="soapenc:string">Formatted WMI Query (1)</flowName>
                     <name xsi:type="soapenc:string">Resolved : success</name>
                     <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                     <startTime xsi:type="xsd:long">1349078108000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
               </steps>
            </runStatusEx>
            <statusCursor xsi:type="ns2:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">0</cursorPosition>
            </statusCursor>
            <statusOptions xsi:type="ns2:WSStatusOptionsEx">
               <primaryResultWanted xsi:type="xsd:boolean">true</primaryResultWanted>
               <rawResultWanted xsi:type="xsd:boolean">false</rawResultWanted>
            </statusOptions>
         </runFlowExReturn>
      </ns1:runFlowExResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getFlowRunHistory

This method retrieves a flow's run history:

```
WSRunHistoryDetails[] getFlowRunHistory(WSRunHistoryQueryParams query)
```

```
WSRunHistoryDetails {
       String userId
       String runStatus
       String flowRevision
       int numSteps
       long runID
       long startTime
       long runDuration
}
```

- `userId` is the name of the user who started the run.
- `runStatus` is the status of the run.
- `flowRevision` is the version of the flow.
- `numSteps` is the number of steps that were executed.
- `runID` is the history run ID from the run history.
- `startTime` is the time of the last flow run since 1/1/1970, in milliseconds.
- `runDuration` is the duration of the run, in milliseconds.

```
WSRunHistoryQueryParams {
       String flowUuid
       long startTime
       long endTime
       int startAtRowIndex
}
```

- `flowUUid` is the unique identifier of the flow.
- `startTime` is the start time, in milliseconds, since 1/1/1970.
- `endTime` is the end time, in milliseconds, since 1/1/1970.
- `startAtRowIndex` is the index from the database specifying the starting point for the available rows retrieval.

Inputs

Input	Type	Description
query	WSRunHistoryQueryParams	The information on the runs, used as filter for the run history.

Outputs

Output	Type	Description
Result	WSRunHistoryDetails[]	Details about the history of the run.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getFlowRunHistory
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <query xsi:type="soap:WSRunHistoryQueryParams"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <endTime xsi:type="xsd:long">1349164381000</endTime>
            <flowUuid xsi:type="xsd:string">b299da5b-280c-4191-8da2-39d3b51d43d7</flowUuid>
            <startAtRowIndex xsi:type="xsd:int">10</startAtRowIndex>
            <startTime xsi:type="xsd:long">0</startTime>
         </query>
      </wsc:getFlowRunHistory>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getFlowRunHistoryResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getFlowRunHistoryReturn soapenc:arrayType="ns2:WSRunHistoryDetails[5]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getFlowRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">131</runId>
               <startTime xsi:type="xsd:long">1349163090000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowRunHistoryReturn>
            <getFlowRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">132</runId>
               <startTime xsi:type="xsd:long">1349163159000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowRunHistoryReturn>
            <getFlowRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">133</runId>
               <startTime xsi:type="xsd:long">1349163184000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowRunHistoryReturn>
            <getFlowRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
```

```
            <duration xsi:type="xsd:long">0</duration>
            <flowRevision xsi:type="soapenc:string">2</flowRevision>
            <numSteps xsi:type="xsd:int">8</numSteps>
            <runId xsi:type="xsd:long">134</runId>
            <startTime xsi:type="xsd:long">1349163188000</startTime>
            <status xsi:type="soapenc:string">Resolved</status>
            <userId xsi:type="soapenc:string">admin</userId>
         </getFlowRunHistoryReturn>
         <getFlowRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
            <duration xsi:type="xsd:long">0</duration>
            <flowRevision xsi:type="soapenc:string">2</flowRevision>
            <numSteps xsi:type="xsd:int">8</numSteps>
            <runId xsi:type="xsd:long">135</runId>
            <startTime xsi:type="xsd:long">1349163189000</startTime>
            <status xsi:type="soapenc:string">Resolved</status>
            <userId xsi:type="soapenc:string">admin</userId>
         </getFlowRunHistoryReturn>
      </getFlowRunHistoryReturn>
   </ns1:getFlowRunHistoryResponse>
</soapenv:Body>
</soapenv:Envelope>
```

getFlowRunHistoryByRunId

This method retrieves a flow's run history based on the value of the `runID`:

```
WSRunHistoryDetailsExtend[] getFlowRunHistoryByRunId(long runID)
```

```
WSRunHistoryDetailsExtend {
       String userId
       String runStatus
       String flowRevision
       int numSteps
       long runID
       long startTime
       long runDuration
       String runName
       String scheduledBy
       String uuid
}
```

- `userId` is the name of the user who started the run.
- `runStatus` is the status of the run.
- `flowRevision` is the version of the flow.
- `numSteps` is the number of steps that were executed.
- `runID` is the history run ID from the run history.
- `startTime` is the time of the last flow run since 1/1/1970, in milliseconds.
- `runDuration` is the duration of the run, in milliseconds.
- `runName` is the name of the flow run.
- `scheduledBy` is the ID of the user who scheduled the flow.
- `uuid` is the UUID of the flow.

Inputs

Input	Type	Description
runID	long	The run identifier of the flow for which to get the history details.

Outputs

Output	Type	Description
Result	WSRunHistoryDetailsExtend[]	Details about the history of the run.

Note: The output is empty if the `runID` used in the request does not exist or if the state of the flow is either is running, stopped, input required or orphaned.

Exceptions

`AxisFault` is thrown if:

- Authentication fails.
- The authenticated user doesn't have `HEADLESS_FLOW` or `RUN_REPORTS` capabilities.
- The `runID` input is empty.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getFlowRunHistoryByRunId
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runId xsi:type="soapenc:long"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">run_ID</runId>
      </wsc:getFlowRunHistoryByRunId>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getFlowRunHistoryByRunIdResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getFlowRunHistoryByRunIdReturn soapenc:arrayType="ns2:WSRunHistoryDetailsExtend[1]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getFlowRunHistoryByRunIdReturn xsi:type="ns2:WSRunHistoryDetailsExtend">
               <duration xsi:type="xsd:long">83</duration>
               <flowRevision xsi:type="soapenc:string">6</flowRevision>
               <numSteps xsi:type="xsd:int">23</numSteps>
               <runId xsi:type="xsd:long">1485</runId>
               <runName xsi:type="soapenc:string">GeFileFolderListTestProperties_
1486</runName>
```

```
                <scheduledBy xsi:type="soapenc:string" xsi:nil="true"/>
                <startTime xsi:type="xsd:long">1357647297361</startTime>
                <status xsi:type="soapenc:string">Resolved</status>
                <userId xsi:type="soapenc:string">admin</userId>
                <uuid xsi:type="soapenc:string">1a92d74b-f759-4b83-a489-5c83a437f35c</uuid>
            </getFlowRunHistoryByRunIdReturn>
         </getFlowRunHistoryByRunIdReturn>
      </ns1:getFlowRunHistoryByRunIdResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getFlowsRunHistory

This method retrieves the run history of an array of flows.

```
WSRunHistoryDetails[] getFlowsRunHistory(WSRunHistoryQueryParams []queries)
```

```
WSRunHistoryDetails {
       String userId
       String runStatus
       String flowRevision
       int numSteps
       long runID
       long startTime
       long runDuration
}
```

- `userId` is the name of the user who started the run.
- `runStatus` is the status of the run.
- `flowRevision` is the version of the flow.
- `numSteps` is the number of steps that were executed.
- `runID` is the history run ID from the run history.
- `startTime` is the time of the last flow run since 1/1/1970, in milliseconds.
- `runDuration` is the duration of the run, in milliseconds.

```
WSRunHistoryQueryParams {
       String flowUuid
       long startTime
       long endTime
       int startAtRowIndex

}
```

- `flowUUid` is the unique identifier of the flow.
- `startTime` is the start time, in milliseconds, since 1/1/1970.
- `endTime` is the end time, in milliseconds, since 1/1/1970.
- `startAtRowIndex` is the index from the database specifying the starting point for the available rows retrieval.

Inputs

Input	Type	Description

queries	WSRunHistoryQueryParams []	The information on the runs, used as filter for the run history.

Outputs

Output	Type	Description
Result	WSRunHistoryDetails []	An array containing details about the history of each run in the query.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getFlowsRunHistory
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <queries xsi:type="wsc:ArrayOf_tns2_WSRunHistoryQueryParams"
soapenc:arrayType="soap:WSRunHistoryQueryParams[]"
xmlns:wsc="http://localhost:8086/PAS/services/WSCentralService"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
       <query xsi:type="soap:WSRunHistoryQueryParams"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <endTime xsi:type="xsd:long">1349164381000</endTime>
            <flowUuid xsi:type="xsd:string">b299da5b-280c-4191-8da2-39d3b51d43d7</flowUuid>
            <startAtRowIndex xsi:type="xsd:int">10</startAtRowIndex>
            <startTime xsi:type="xsd:long">0</startTime>
        </query>
       <query xsi:type="soap:WSRunHistoryQueryParams"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <endTime xsi:type="xsd:long">1349164381000</endTime>
            <flowUuid xsi:type="xsd:string">d012e1c3-704f-426f-a380-b2425a166d39</flowUuid>
            <startAtRowIndex xsi:type="xsd:int">0</startAtRowIndex>
            <startTime xsi:type="xsd:long">0</startTime>
        </query>
       </queries>
      </wsc:getFlowsRunHistory>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
<ns1:getFlowsRunHistoryResponse
```

```
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1=http://wscentralservice.services.dharma.iconclude.com>
         <getFlowsRunHistoryReturn soapenc:arrayType="ns2:WSRunHistoryDetails[13]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">131</runId>
               <startTime xsi:type="xsd:long">1349163090000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowsRunHistoryReturn>
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">132</runId>
               <startTime xsi:type="xsd:long">1349163159000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowsRunHistoryReturn>
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">133</runId>
               <startTime xsi:type="xsd:long">1349163184000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowsRunHistoryReturn>
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">134</runId>
               <startTime xsi:type="xsd:long">1349163188000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowsRunHistoryReturn>
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">0</duration>
               <flowRevision xsi:type="soapenc:string">2</flowRevision>
               <numSteps xsi:type="xsd:int">8</numSteps>
               <runId xsi:type="xsd:long">135</runId>
               <startTime xsi:type="xsd:long">1349163189000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowsRunHistoryReturn>
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">5</duration>
               <flowRevision xsi:type="soapenc:string">30</flowRevision>
               <numSteps xsi:type="xsd:int">12</numSteps>
               <runId xsi:type="xsd:long">87</runId>
               <startTime xsi:type="xsd:long">1348659298000</startTime>
               <status xsi:type="soapenc:string">Resolved</status>
               <userId xsi:type="soapenc:string">admin</userId>
            </getFlowsRunHistoryReturn>
            <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
               <duration xsi:type="xsd:long">1</duration>
```

```
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">90</runId>
      <startTime xsi:type="xsd:long">1348659391000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
   <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
      <duration xsi:type="xsd:long">1</duration>
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">93</runId>
      <startTime xsi:type="xsd:long">1348659447000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
   <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
      <duration xsi:type="xsd:long">1</duration>
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">97</runId>
      <startTime xsi:type="xsd:long">1348660057000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
   <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
      <duration xsi:type="xsd:long">5</duration>
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">103</runId>
      <startTime xsi:type="xsd:long">1348660460000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
   <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
      <duration xsi:type="xsd:long">12</duration>
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">114</runId>
      <startTime xsi:type="xsd:long">1348675467000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
   <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
      <duration xsi:type="xsd:long">12</duration>
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">118</runId>
      <startTime xsi:type="xsd:long">1348677815000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
   <getFlowsRunHistoryReturn xsi:type="ns2:WSRunHistoryDetails">
      <duration xsi:type="xsd:long">10</duration>
      <flowRevision xsi:type="soapenc:string">30</flowRevision>
      <numSteps xsi:type="xsd:int">12</numSteps>
      <runId xsi:type="xsd:long">121</runId>
      <startTime xsi:type="xsd:long">1348677840000</startTime>
      <status xsi:type="soapenc:string">Resolved</status>
      <userId xsi:type="soapenc:string">admin</userId>
   </getFlowsRunHistoryReturn>
```

```
        </getFlowsRunHistoryReturn>
      </ns1:getFlowsRunHistoryResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getRunStatus

This method retrieves the status of a run:

```
WSRunStatus getRunStatus(WSRunHandle runHandle)
```

```
WSRunStatus {
       WSRunStepDetails[] steps
       String runResponse
       String runUrl
       String runReportUrl
       WSRunHandle runHandle
       int status
}
WSRunStepDetails {
       String name
       String flowName
       String stepResonse
       Long startTime
       Long endTime
       Int runStepLevel
       WSRunHandle[] childRuns

}
WSRunHandle {
       String runID
       WSRunStatusCursor statusCursor
}
WSRunStatusCursor {
       int cursorPosition
}
```

Inputs

Input	Type	Description
runHandle	WSRunHandle	The information on the run: the run ID and the starting position in the step history.

Outputs

Output	Type	Description
Result	WSRunStatus	The result includes: • The response of the run. • The URL of the run report. • The URL of the run. • The run ID and the current step position. • The status of the run. • The step details.

The available run status values are:

- Unknown = -1
- Running = 0
- Paused = 1
- Ended/Finished = 2
- Canceled = 3

The step details information includes:

- The name of the step.
- The name of the flow or subflow containing the step.
- The step response.
- The step start time, in milliseconds, since 1/1/1970.
- The step end time, in milliseconds, since 1/1/1970.
- A `WSRunHandles` array representing the parallel child runs of the current step. The step levels for these parallel runs start at `1`.
- The run step level. This is a zero based integer, representing the step level in the flow run. Each subflow run represents a level. For example, if the client runs **flowA**, which runs **flowB**, which in turn runs **flowC**, the level of all the steps under **flowC** is `3`. Level `0` is the flow itself.

Exceptions

`AxisFault` is thrown if:

- `runHandle` is null.
- The provided run ID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getRunStatus soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runHandle xsi:type="soap:WSRunHandle"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">661</runID>
            <statusCursor xsi:type="soap:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">0</cursorPosition>
            </statusCursor>
         </runHandle>
      </wsc:getRunStatus>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getRunStatusResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getRunStatusReturn xsi:type="ns2:WSRunStatus"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <runHandle xsi:type="ns2:WSRunHandle">
               <runID xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">661</runID>
               <statusCursor xsi:type="ns2:WSRunStatusCursor">
                  <cursorPosition xsi:type="xsd:int">2</cursorPosition>
               </statusCursor>
            </runHandle>
            <runReportUrl xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">https://my-
ooserver.myco.com:8443/PAS/app?service=RCLinkService/ReportLinkDispatch&amp;sp=SINDIVIDUAL_
REPAIR_LEVEL&amp;sp=S31ea143f-2ae5-41bf-b44b-
7a8c423011c5&amp;sp=l0&amp;sp=l12890&amp;sp=l661</runReportUrl>
            <runResponse xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
            <runResumeUrl xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">https://my-
ooserver.myco.com:8443/PAS-
/app?service=RCLinkService/FlowLinkDispatch&amp;sp=SRESUME&amp;sp=l661</runResumeUrl>
            <status xsi:type="xsd:int">1</status>
            <steps soapenc:arrayType="ns2:WSRunStepDetails[2]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]" xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1348746340000</endTime>
                  <flowName xsi:type="soapenc:string">Get Stopped Service List</flowName>
                  <name xsi:type="soapenc:string">Add List To Global Context</name>
                  <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                  <startTime xsi:type="xsd:long">1348746340000</startTime>
                  <stepResponse xsi:type="soapenc:string">success</stepResponse>
               </steps>
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]" xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1348746340000</endTime>
                  <flowName xsi:type="soapenc:string">Get Stopped Service List</flowName>
                  <name xsi:type="soapenc:string">Resolved : success</name>
                  <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                  <startTime xsi:type="xsd:long">1348746340000</startTime>
                  <stepResponse xsi:type="soapenc:string">success</stepResponse>
               </steps>
            </steps>
         </getRunStatusReturn>
      </ns1:getRunStatusResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getRunStatusEx

This method returns the run results: the primary result and the raw result.

```
WSRunStatusEx getRunStatusEx(WSRunHandle runHandle, WSStatusOptionsEx options)
```

```
WSRunStatusEx {
       String primaryResult
       String rawResult
       WSStatusOptionsEx statusOptions
}
WSStatusOptionsEx {
       boolean rawResultWanted
       boolean primaryResultWanted
}
WSRunHandle {
       String runID
       WSRunStatusCursor statusCursor
}
WSRunStatusCursor {
       int cursorPosition
}
```

Inputs

Input	Type	Description
runHandle	WSRunHandle	The information on the run: the run ID and the starting position in the step history.
options	WSStatusOptionsEx	The type of result that will be returned by this method: primary and/or raw.

Outputs

Output	Type	Description
Result	WSRunStatusEx	The result includes: • The primary result. • The raw result. • The options included in the get status call.

Exceptions

`AxisFault` is thrown if:

- `runHandle` is null.
- The provided run ID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <wsc:getRunStatusEx soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runHandle xsi:type="soap:WSRunHandle"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">661</runID>
            <statusCursor xsi:type="soap:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">5</cursorPosition>
            </statusCursor>
         </runHandle>
         <options xsi:type="soap:WSStatusOptionsEx"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <primaryResultWanted xsi:type="xsd:boolean">true</primaryResultWanted>
            <rawResultWanted xsi:type="xsd:boolean">true</rawResultWanted>
         </options>
      </wsc:getRunStatusEx>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getRunStatusExResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getRunStatusExReturn xsi:type="ns2:WSRunStatusEx"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <primaryResult xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">[Application Layer Gateway
Service;Application Identity;Application Information;Application Management;ASP.NET State
Service;Windows Audio Endpoint Builder;Windows Audio;Background Intelligent Transfer
Service;Microsoft .NET Framework NGEN v4.0.30319_X86;Microsoft .NET Framework NGEN v4.0.30319_
X64;Cluster Service;COM+ System Application;Disk Defragmenter;Wired AutoConfig;Extensible
Authentication Protocol;Encrypting File System (EFS);Microsoft Fibre Channel Platform
Registration Service;Function Discovery Provider Host;Function Discovery Resource
Publication;Windows Font Cache Service;Windows Presentation Foundation Font Cache
3.0.0.0;Human Interface Device Access;Health Key and Certificate Management;Windows
CardSpace;CNG Key Isolation;KtmRm for Distributed Transaction Coordinator;Link-Layer Topology
Discovery Mapper;Multimedia Class Scheduler;Windows Installer;SQL Server
(MSSQLSERVER);MySQL;Network Access Protection Agent;Netlogon;OpenVPN Service;Performance
Counter DLL Host;Performance Logs &amp;amp; Alerts;Protected Storage;Remote Access Auto Connection
Manager;Remote Access Connection Manager;Remote Procedure Call (RPC) Locator;Resultant Set of
Policy Provider;Special Administration Console Helper;Smart Card;Smart Card Removal
Policy;Secondary Logon;Internet Connection Sharing (ICS);Shell Hardware Detection;SNMP
Trap;Software Protection;SPP Notification Service;SQL Server Agent (MSSQLSERVER);Secure Socket
Tunneling Protocol Service;Microsoft Software Shadow Copy Provider;Telephony;TPM Base
Services;Thread Ordering Server;Windows Modules Installer;Interactive Services
Detection;Credential Manager;Virtual Disk;Volume Shadow Copy;Windows Process Activation
Service;Windows Color System;Diagnostic Service Host;Diagnostic System Host;Windows Event
Collector;Problem Reports and Solutions Control Panel Support;Windows Error Reporting
Service;WinHTTP Web Proxy Auto-Discovery Service;WMI Performance Adapter;Portable Device
Enumerator Service;Windows Driver Foundation - User-mode Driver Framework]</primaryResult>
            <rawResult xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">{Field 1=services;Field 2=
[Application Layer Gateway Service;Application Identity;Application Information;Application
Management;ASP.NET State Service;Windows Audio Endpoint Builder;Windows Audio;Background
Intelligent Transfer Service;Microsoft .NET Framework NGEN v4.0.30319_X86;Microsoft .NET
Framework NGEN v4.0.30319_X64;Cluster Service;COM+ System Application;Disk Defragmenter;Wired
AutoConfig;Extensible Authentication Protocol;Encrypting File System (EFS);Microsoft Fibre
Channel Platform Registration Service;Function Discovery Provider Host;Function Discovery
Resource Publication;Windows Font Cache Service;Windows Presentation Foundation Font Cache
```

3.0.0.0;Human Interface Device Access;Health Key and Certificate Management;Windows CardSpace;CNG Key Isolation;KtmRm for Distributed Transaction Coordinator;Link-Layer Topology Discovery Mapper;Multimedia Class Scheduler;Windows Installer;SQL Server (MSSQLSERVER);MySQL;Network Access Protection Agent;Netlogon;OpenVPN Service;Performance Counter DLL Host;Performance Logs & Alerts;Protected Storage;Remote Access Auto Connection Manager;Remote Access Connection Manager;Remote Procedure Call (RPC) Locator;Resultant Set of Policy Provider;Special Administration Console Helper;Smart Card;Smart Card Removal Policy;Secondary Logon;Internet Connection Sharing (ICS);Shell Hardware Detection;SNMP Trap;Software Protection;SPP Notification Service;SQL Server Agent (MSSQLSERVER);Secure Socket Tunneling Protocol Service;Microsoft Software Shadow Copy Provider;Telephony;TPM Base Services;Thread Ordering Server;Windows Modules Installer;Interactive Services Detection;Credential Manager;Virtual Disk;Volume Shadow Copy;Windows Process Activation Service;Windows Color System;Diagnostic Service Host;Diagnostic System Host;Windows Event Collector;Problem Reports and Solutions Control Panel Support;Windows Error Reporting Service;WinHTTP Web Proxy Auto-Discovery Service;WMI Performance Adapter;Portable Device Enumerator Service;Windows Driver Foundation - User-mode Driver Framework];Result=[Application Layer Gateway Service;Application Identity;Application Information;Application Management;ASP.NET State Service;Windows Audio Endpoint Builder;Windows Audio;Background Intelligent Transfer Service;Microsoft .NET Framework NGEN v4.0.30319_X86;Microsoft .NET Framework NGEN v4.0.30319_X64;Cluster Service;COM+ System Application;Disk Defragmenter;Wired AutoConfig;Extensible Authentication Protocol;Encrypting File System (EFS);Microsoft Fibre Channel Platform Registration Service;Function Discovery Provider Host;Function Discovery Resource Publication;Windows Font Cache Service;Windows Presentation Foundation Font Cache 3.0.0.0;Human Interface Device Access;Health Key and Certificate Management;Windows CardSpace;CNG Key Isolation;KtmRm for Distributed Transaction Coordinator;Link-Layer Topology Discovery Mapper;Multimedia Class Scheduler;Windows Installer;SQL Server (MSSQLSERVER);MySQL;Network Access Protection Agent;Netlogon;OpenVPN Service;Performance Counter DLL Host;Performance Logs & Alerts;Protected Storage;Remote Access Auto Connection Manager;Remote Access Connection Manager;Remote Procedure Call (RPC) Locator;Resultant Set of Policy Provider;Special Administration Console Helper;Smart Card;Smart Card Removal Policy;Secondary Logon;Internet Connection Sharing (ICS);Shell Hardware Detection;SNMP Trap;Software Protection;SPP Notification Service;SQL Server Agent (MSSQLSERVER);Secure Socket Tunneling Protocol Service;Microsoft Software Shadow Copy Provider;Telephony;TPM Base Services;Thread Ordering Server;Windows Modules Installer;Interactive Services Detection;Credential Manager;Virtual Disk;Volume Shadow Copy;Windows Process Activation Service;Windows Color System;Diagnostic Service Host;Diagnostic System Host;Windows Event Collector;Problem Reports and Solutions Control Panel Support;Windows Error Reporting Service;WinHTTP Web Proxy Auto-Discovery Service;WMI Performance Adapter;Portable Device Enumerator Service;Windows Driver Foundation - User-mode Driver Framework];}</rawResult>
            <runHandle xsi:type="ns2:WSRunHandle">
               <runID xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">661</runID>
               <statusCursor xsi:type="ns2:WSRunStatusCursor">
                  <cursorPosition xsi:type="xsd:int">9</cursorPosition>
               </statusCursor>
            </runHandle>
            <runReportUrl xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">https://my-
ooserver.myco.com:8443/PAS/app?service=RCLinkService/ReportLinkDispatch&sp=SINDIVIDUAL_
REPAIR_LEVEL&sp=S31ea143f-2ae5-41bf-b44b-
7a8c423011c5&sp=l0&sp=l12890&sp=l661</runReportUrl>
            <runResponse xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Error</runResponse>
            <runResumeUrl xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
            <status xsi:type="xsd:int">2</status>
            <statusOptions xsi:type="ns2:WSStatusOptionsEx">
               <primaryResultWanted xsi:type="xsd:boolean">true</primaryResultWanted>
               <rawResultWanted xsi:type="xsd:boolean">true</rawResultWanted>
            </statusOptions>
            <steps soapenc:arrayType="ns2:WSRunStepDetails[4]" xsi:type="soapenc:Array"

```
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]" xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1348746340000</endTime>
                  <flowName xsi:type="soapenc:string">Get Stopped Service List</flowName>
                  <name xsi:type="soapenc:string">Add List To Global Context</name>
                  <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                  <startTime xsi:type="xsd:long">1348746340000</startTime>
                  <stepResponse xsi:type="soapenc:string">success</stepResponse>
               </steps>
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]" xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1348746340000</endTime>
                  <flowName xsi:type="soapenc:string">Get Stopped Service List</flowName>
                  <name xsi:type="soapenc:string">Resolved : success</name>
                  <runStepLevel xsi:type="xsd:int">2</runStepLevel>
                  <startTime xsi:type="xsd:long">1348746340000</startTime>
                  <stepResponse xsi:type="soapenc:string">success</stepResponse>
               </steps>
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]" xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1348747771000</endTime>
                  <flowName xsi:type="soapenc:string">Restart Service - Tutorial
Flow</flowName>
                  <name xsi:type="soapenc:string">Select a Service</name>
                  <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                  <startTime xsi:type="xsd:long">1348747766000</startTime>
                  <stepResponse xsi:type="soapenc:string">failure</stepResponse>
               </steps>
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]" xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1348747771000</endTime>
                  <flowName xsi:type="soapenc:string">Restart Service - Tutorial
Flow</flowName>
                  <name xsi:type="soapenc:string">Error : failure</name>
                  <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                  <startTime xsi:type="xsd:long">1348747771000</startTime>
                  <stepResponse xsi:type="soapenc:string">failure</stepResponse>
               </steps>
            </steps>
         </getRunStatusExReturn>
      </ns1:getRunStatusExResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getStatusForRuns

This method returns the runs' status:

```
WSRunStatus[] getStatusForRuns(WSRunHandle[] runHandles)
```

```
WSRunStatus {
       WSRunStepDetails[] steps
       String runResponse
       String runUrl
       String runReportUrl
       WSRunHandle runHandle
       int status
}
WSRunStepDetails {
```

```
        String name
        String flowName
        String stepResonse
        Long startTime
        Long endTime
        Int runStepLevel
        WSRunHandle[] childRuns

}
WSRunHandle {
        String runID
        WSRunStatusCursor statusCursor
}
WSRunStatusCursor {
        int cursorPosition
}
```

Inputs

Input	Type	Description
runHandles	WSRunHandle []	An array of structures containing the information on the runs: the run IDs and the starting position in the step history.

Outputs

Output	Type	Description
Result	WSRunStatus[]	For each run, the result includes: • The response of the run. • The URL of the run report. • The URL of the run. • The run ID and the current step position. • The status of the run. • The step details.

The available run status values are:

- Unknown = -1
- Running = 0
- Paused = 1
- Ended/Finished = 2
- Canceled = 3

The step details information includes:

- The name of the step.
- The name of the flow or subflow containing the step.

- The step response.
- The step start time, in milliseconds, since 1/1/1970.
- The step end time, in milliseconds, since 1/1/1970.
- A `WSRunHandles` array representing the parallel child runs of the current step. The step levels for these parallel runs start at `1`.
- The run step level. This is a zero based integer, representing the step level in the flow run. Each subflow run represents a level. For example, if the client runs **flowA**, which runs **flowB**, which in turn runs **flowC**, the level of all the steps under **flowC** is `3`. Level `0` is the flow itself.

Exceptions

`AxisFault` is thrown if:

- `runHandle` is null.
- The provided run ID is not valid.
- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getStatusForRuns soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <runHandles xsi:type="wsc:ArrayOf_tns2_WSRunHandle"
soapenc:arrayType="soap:WSRunHandle[]" xmlns:wsc="https://my-
ooserver.myco.com:8443/PAS/services/WSCentralService"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
       <runHandle xsi:type="soap:WSRunHandle"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">30633</runID>
            <statusCursor xsi:type="soap:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">0</cursorPosition>
            </statusCursor>
         </runHandle>
       <runHandle xsi:type="soap:WSRunHandle"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">30629</runID>
            <statusCursor xsi:type="soap:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">3</cursorPosition>
            </statusCursor>
         </runHandle>
       <runHandle xsi:type="soap:WSRunHandle"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <runID xsi:type="xsd:string">30581</runID>
            <statusCursor xsi:type="soap:WSRunStatusCursor">
               <cursorPosition xsi:type="xsd:int">0</cursorPosition>
            </statusCursor>
         </runHandle>
```

```
       </runHandles>
      </wsc:getStatusForRuns>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getStatusForRunsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getStatusForRunsReturn soapenc:arrayType="ns2:WSRunStatus[3]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getStatusForRunsReturn xsi:type="ns2:WSRunStatus">
               <runHandle xsi:type="ns2:WSRunHandle">
                  <runID xsi:type="soapenc:string">30633</runID>
                  <statusCursor xsi:type="ns2:WSRunStatusCursor">
                     <cursorPosition xsi:type="xsd:int">3</cursorPosition>
                  </statusCursor>
               </runHandle>
               <runReportUrl xsi:type="soapenc:string">https://my-
ooserver.myco.com:8443/PAS/app?service=RCLinkService/ReportLinkDispatch&amp;sp=SINDIVIDUAL_
REPAIR_LEVEL&amp;sp=S602fdf9a-be77-4600-83fd-
f7a5ad56a85e&amp;sp=l0&amp;sp=l42862&amp;sp=l30633</runReportUrl>
               <runResponse xsi:type="soapenc:string">Resolved</runResponse>
               <runResumeUrl xsi:type="soapenc:string"/>
               <status xsi:type="xsd:int">2</status>
               <steps soapenc:arrayType="ns2:WSRunStepDetails[3]" xsi:type="soapenc:Array">
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349073360000</endTime>
                     <flowName xsi:type="soapenc:string">myTest</flowName>
                     <name xsi:type="soapenc:string">myTest</name>
                     <runStepLevel xsi:type="xsd:int">0</runStepLevel>
                     <startTime xsi:type="xsd:long">1349073360000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349073360000</endTime>
                     <flowName xsi:type="soapenc:string">myTest</flowName>
                     <name xsi:type="soapenc:string">Random Number Generator</name>
                     <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                     <startTime xsi:type="xsd:long">1349073360000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                  <steps xsi:type="ns2:WSRunStepDetails">
                     <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                     <endTime xsi:type="xsd:long">1349073360000</endTime>
                     <flowName xsi:type="soapenc:string">myTest</flowName>
                     <name xsi:type="soapenc:string">Resolved : success</name>
                     <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                     <startTime xsi:type="xsd:long">1349073360000</startTime>
                     <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
```

```
            </steps>
         </getStatusForRunsReturn>
         <getStatusForRunsReturn xsi:type="ns2:WSRunStatus">
            <runHandle xsi:type="ns2:WSRunHandle">
               <runID xsi:type="soapenc:string">30629</runID>
               <statusCursor xsi:type="ns2:WSRunStatusCursor">
                  <cursorPosition xsi:type="xsd:int">3</cursorPosition>
               </statusCursor>
            </runHandle>
            <runReportUrl xsi:type="soapenc:string">https://my-
ooserver.myco.com:8443/PAS/app?service=RCLinkService/ReportLinkDispatch&amp;sp=SINDIVIDUAL_
REPAIR_LEVEL&amp;sp=S602fdf9a-be77-4600-83fd-
f7a5ad56a85e&amp;sp=l0&amp;sp=l42858&amp;sp=l30629</runReportUrl>
            <runResponse xsi:type="soapenc:string">Resolved</runResponse>
            <runResumeUrl xsi:type="soapenc:string"/>
            <status xsi:type="xsd:int">2</status>
            <steps soapenc:arrayType="ns2:WSRunStepDetails[0]" xsi:type="soapenc:Array"/>
         </getStatusForRunsReturn>
         <getStatusForRunsReturn xsi:type="ns2:WSRunStatus">
            <runHandle xsi:type="ns2:WSRunHandle">
               <runID xsi:type="soapenc:string">30581</runID>
               <statusCursor xsi:type="ns2:WSRunStatusCursor">
                  <cursorPosition xsi:type="xsd:int">3</cursorPosition>
               </statusCursor>
            </runHandle>
            <runReportUrl xsi:type="soapenc:string">https://my-
ooserver.myco.com:8443/PAS/app?service=RCLinkService/ReportLinkDispatch&amp;sp=SINDIVIDUAL_
REPAIR_LEVEL&amp;sp=S602fdf9a-be77-4600-83fd-
f7a5ad56a85e&amp;sp=l0&amp;sp=l42810&amp;sp=l30581</runReportUrl>
            <runResponse xsi:type="soapenc:string">Resolved</runResponse>
            <runResumeUrl xsi:type="soapenc:string"/>
            <status xsi:type="xsd:int">2</status>
            <steps soapenc:arrayType="ns2:WSRunStepDetails[3]" xsi:type="soapenc:Array">
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1349072880000</endTime>
                  <flowName xsi:type="soapenc:string">myTest</flowName>
                  <name xsi:type="soapenc:string">myTest</name>
                  <runStepLevel xsi:type="xsd:int">0</runStepLevel>
                  <startTime xsi:type="xsd:long">1349072880000</startTime>
                  <stepResponse xsi:type="soapenc:string">success</stepResponse>
               </steps>
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1349072880000</endTime>
                  <flowName xsi:type="soapenc:string">myTest</flowName>
                  <name xsi:type="soapenc:string">Random Number Generator</name>
                  <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                  <startTime xsi:type="xsd:long">1349072880000</startTime>
                  <stepResponse xsi:type="soapenc:string">success</stepResponse>
               </steps>
               <steps xsi:type="ns2:WSRunStepDetails">
                  <childRuns soapenc:arrayType="ns2:WSRunHandle[0]"
xsi:type="soapenc:Array"/>
                  <endTime xsi:type="xsd:long">1349072880000</endTime>
                  <flowName xsi:type="soapenc:string">myTest</flowName>
                  <name xsi:type="soapenc:string">Resolved : success</name>
                  <runStepLevel xsi:type="xsd:int">1</runStepLevel>
                  <startTime xsi:type="xsd:long">1349072880000</startTime>
```

```
                    <stepResponse xsi:type="soapenc:string">success</stepResponse>
                  </steps>
                </steps>
              </getStatusForRunsReturn>
            </getStatusForRunsReturn>
          </ns1:getStatusForRunsResponse>
        </soapenv:Body>
      </soapenv:Envelope>
```

Scheduler

The classes and the methods documented in this section enable you to handle the OO Central Scheduler.

Classes:

- "ScheduleDisplayInfo" on page 236.
- "Pair" on page 238.
- "ScheduledFlowInfo" on page 240.
- "ScheduleInfo" below.

Methods:

- "scheduleFlow" on page 248.
- "getSchedule" on page 251.
- "pauseSchedule" on page 255.
- "isSchedulePaused" on page 255.
- "resumeSchedule" on page 256.
- "deleteSchedule" on page 257.
- "isSchedulerEnabled" on page 258.
- "getScheduledFlows" on page 259.
- "getSchedulesOfFlow" on page 261.
- "getSchedulesForFlowCategory" on page 263.
- "pauseScheduledFlow" on page 266.
- "isScheduledFlowPaused" on page 267.
- "resumeScheduledFlow" on page 268.
- "deleteScheduledFlow" on page 269.

Classes for working with Scheduler

ScheduleInfo

```
com.iconclude.dharma.scheduler.web

Class ScheduleInfo

java.lang.Object
               |-com.iconclude.dharma.scheduler.web.ScheduleInfo
```

All Implemented Interfaces:

java.io.Serializable

```
public class ScheduleInfo
extends java.lang.Object
implements java.io.Serializable
```

This object holds the scheduled trigger information,the flow name and flow parameters, if needed.

Serialized Fields

description

```
java.lang.String description
```

---

enabled

```
java.lang.Boolean enabled
```

---

endTime

```
java.util.Calendar endTime
```

---

name

```
java.lang.String name
```

---

params

```
Pair[] params
```

---

repeatCount

```
int repeatCount
```

---

repeatIntervalMilli

```
long repeatIntervalMilli
```

---

startTime

```
java.util.Calendar startTime
```

---

units

```
java.lang.String units
```

---

type

```
int type
```

---

executing

```
java.lang.Boolean executing
```

---

nextRuntime

```
java.util.Calendar nextRuntime
```

---

prevRuntime

```
java.util.Calendar prevRuntime
```

---

cronExpression

```
java.lang.String cronExpression
```

---

dayNumber

```
int dayNumber
```

---

monthNumber

```
int monthNumber
```

---

dayType

```
int dayType
```

---

dayOrder

```
int dayOrder
```

---

paused

```
java.lang.Boolean paused
```

---

triggerName

```
java.lang.String triggerName
```

Constructor Summary

```
ScheduleInfo()
```

```
ScheduleInfo (java.lang.String description, java.lang.Boolean enabled,
java.util.Calendar endTime, java.lang.String name, Pair[] params, int
repeatCount, long repeatIntervalMilli, java.util.Calendar startTime,
java.lang.String units, int type, java.lang.Boolean executing,
java.util.Calendar nextRuntime, java.util.Calendar prevRuntime,
java.lang.String cronExpression, int dayNumber, int monthNumber, int
dayType, int dayOrder, java.lang.String triggerName, java.lang.Boolean
paused)
```

Method Summary

The following table contains the method names, their type and a description of their effect on `ScheduleInfo`.

Type	Method	What it does
`java.lang.String`	`getCronExpression()`	Gets the cronExpression value.
`int`	`getDayNumber()`	Gets the dayNumber value.
`int`	`getDayOrder()`	Gets the dayOrder value.
`int`	`getDayType()`	Gets the dayType value.
`java.lang.String`	`getDescription()`	Gets the description value.
`java.lang.Boolean`	`getEnabled()`	Gets the enabled value.
`java.util.Calendar`	`getEndTime()`	Gets the endTime value.
`java.lang.Boolean`	`getExecuting()`	Gets the executing value.
`ing`	`getMonthNumber()`	Gets the monthNumber value.
`java.lang.String`	`getName()`	Gets the name value.
`java.util.Calendar`	`getNextRuntime()`	Gets the nextRuntime value.
`Pair`	`getParams()`	Gets the params value.
`java.lang.Boolean`	`getPaused()`	Returns whether the schedule state is `PAUSED` or not.
`java.util.Calendar`	`getPrevRuntime()`	Gets the prevRuntime value.

Type	Method	What it does
`int`	`getRepeatCount()`	Gets the repeatCount value.
`long`	`getRepeatIntervalMilli()`	Gets the repeatIntervalMilli value.
`java.util.Calendar`	`getStartTime()`	Gets the StartTime value.
`java.lang.String`	`getTriggerName()`	Gets the triggerName value.
`int`	`getType()`	Gets the type value.
`java.lang.String`	`getUnits()`	Gets the units value.
`void`	`setCronExpression (java.lang.String cronExpression)`	Sets the schedule trigger information using the `quartz con` expression String format (currently not supported).
`void`	`setDayNumber(int dayNumber)()`	Sets the dayNumber value.
`void`	`setDayOrder(int dayOrder)`	Sets the dayOrder value .Used for `TYPE _MONTHLY` and `TYPE_ YEARLY`.
`void`	`setDayType(int dayType)`	Sets the dayType value.
`void`	`setDescription (java.lang.String description)`	Sets the description value.
`void`	`setEnabled (java.lang.Boolean enabled)`	Set `false` to create the schedule disabled. Otherwise, set to `true`.
`void`	`setEndTime (java.util.Calendar endTime)`	Sets the endTime value.
`void`	`setExecuting (java.lang.Boolean executing)`	Sets the executing value.
`void`	`setMonthNumber(int monthNumber)`	Sets the monthNumber value.
`void`	`setName (java.lang.String name)`	Sets the name of the created schedule.

Type	Method	What it does
void	setNextRuntime (java.util.Calendar nextRuntime)	Sets the nextRuntime value.
void	setParams(Pair[] params)	Sets the params value for flow which is to be invoked by the schedule.
void	setPaused (java.lang.Boolean paused)	Not relevant as input.
void	setPrevRuntime (java.util.Calendar prevRuntime)	Sets the prevRuntime value.
void	setRepeatCount(int repeatCount)	Sets the repeat count value for this Schedule trigger of type Interval only.
void	setRepeatIntervalMilli (long repeatIntervalMilli)	Sets the repeat count value for this Schedule schedule of type TYPE_INTERVAL only.
void	setStartTime (java.util.Calendar startTime)	Sets the endTime value .
void	setTriggerName (java.lang.String triggerName)	Sets the trigger name for the schedule.
void	setType(int type)	Sets the type value.
void	setUnits (java.lang.String units)	Sets the units value.

Constructor Detail

ScheduleInfo

```
public ScheduleInfo()
```

ScheduleInfo

```
public ScheduleInfo(java.lang.String description,
                    java.lang.Boolean enabled,
                    java.util.Calendar endTime,
                    java.lang.String name,
                    Pair[] params,
```

```
                int repeatCount,
                long repeatIntervalMilli,
                java.util.Calendar startTime,
                java.lang.String units,
                int type,
                java.lang.Boolean executing,
                java.util.Calendar nextRuntime,
                java.util.Calendar prevRuntime,
                java.lang.String cronExpression,
                int dayNumber,
                int monthNumber,
                int dayType,
                int dayOrder,
                java.lang.String triggerName,
                java.lang.Boolean paused)
```

Methods Detail

getDescription

```
public java.lang.String getDescription()
```

Gets the description value for this `ScheduleInfo`.

Returns:

description

---

setDescription

```
public void setDescription(java.lang.String description)
```

Sets the description for this Pair.

Parameters:

`description` -

---

getEnabled

```
public java.lang.Boolean getEnabled()
```

Gets the enabled for this ScheduleInfo. The schedule counted as enabled when it is state is NORMAL or BLOCKED:

- NORMAL
- BLOCKED: schedule is currently executing/running

Returns:

return true if schedule state enabled

---

setEnabled

```
public void setEnabled(java.lang.Boolean enabled)
```

Set false to create the schedule disabled otherwise set to true.

Parameters:

`enabled` -

---

getEndTime

```
public java.util.Calendar getEndTime()
```

Gets the endTime for this ScheduleInfo.

Returns:

endTime

---

setEndTime

```
public void setEndTime(java.util.Calendar endTime)
```

Sets the endTime value for this ScheduleInfo. determine the end time the schedule will be stopped from being triggered. When using SOAP UI tools the date time format is “MM/DD/YYYY HH:MM:SS AM/PM” . For example "12/19/2011 9:35:38 AM".

Parameters:

`endTime` - Calendar instance

---

getName

```
public java.lang.String getName()
```

Gets the name for this ScheduleInfo.

Returns:

name

---

setName

```
public void setName(java.lang.String name)
```

Sets the name of the created schedule. If you intended to create a new schedule then the name you choose should not be used before otherwise an already existed schedule will be updated.

Parameters:

```
name -
```

---

getParams

```
public Pair[] getParams()
```

Gets the params value for this ScheduleInfo.

Returns:

```
params
```

---

setParams

```
public void setParams(Pair[] params)
```

Gets the params value for this ScheduleInfo.

Parameters:

```
params -
```

---

getRepeatCount

```
public int getRepeatCount()
```

Gets the repeatCount value for this ScheduleInfo.

Returns:

```
repeatCount
```

---

setRepeatCount

```
public void setRepeatCount(int repeatCount)
```

Sets the repeat count value for this Schedule trigger of type Interval only. The actual repeat times of the created schedule will be the repeatCount value + 1.

Parameters:

`repeatCount -`

---

getRepeatIntervalMilli

```
public long getRepeatIntervalMilli()
```

Gets the repeatIntervalMilli value for this ScheduleInfo.

Returns:

`repeatIntervalMilli`

---

setRepeatIntervalMilli

```
public void setRepeatIntervalMilli(long repeatIntervalMilli)
```

Sets the repeat count value for this Schedule schedule of type ” TYPE_INTERVAL” only. The actual repeat times of the created schedule in scheduler will be the repeatCount value + 1.

Parameters:

`repeatIntervalMilli -`

---

getStartTime

```
public java.util.Calendar getStartTime()
```

Gets the startTime value for this ScheduleInfo.

Returns:

`startTime`

---

setStartTime

```
public void setStartTime(java.util.Calendar startTime)
```

Sets the endTime value for this ScheduleInfo. determine the end time the schedule will be stopped from being triggered. When using SOAP UI tools the date time format is “MM/DD/YYYY HH:MM:SS AM/PM”. For example "12/19/2011 9:35:38 AM".

Parameters:

`startTime` -Calendar instance

---

getUnits

```
public java.lang.String getUnits()
```

Gets the units value for this ScheduleInfo.

Returns:

```
units
```

---

setUnits

```
public void setUnits(java.lang.String units)
```

Sets the units value for this ScheduleInfo. Used for schedule of type “TYPE_INTERVAL”.valid values:”minutes”,"hours" .

Parameters:

```
units -
```

---

getType

```
public int getType()
```

Gets the type value for this ScheduleInfo.

Returns:

type

---

setType

```
public void setType(int type)
```

Sets the type value for this ScheduleInfo. Valid types and the must set schedule info properties:

- `TYPE_OLD_UI = 0`:
  - `repeatIntervalMilli & unit[”minutes”,"hours"]`
- `TYPE_INTERVAL = 1`: make Interval trigger each RepeatIntervalMilli property value
- `TYPE_EVERY_DAY = 2`: make daily trigger
- `TYPE_EVERY_WEEKDAY = 3`: make every week day trigger, only working days - MON-FRI
- `TYPE_WEEKLY = 4`: make weekly trigger, property DayNumber have to be set to one of the week days.

- dayNumber[Sun=1,Mon=2,Tue=4,Wed=8,Thu=16,Fri=32,Sat=64]

- TYPE_MONTHLY = 5: make monthly trigger, property DayNumber have to be set to one of:
  - dayNumber[1 to 31]
  - dayType[0,1,2,3,4,5,6,7] & dayOrder[0,1,2,3,4]
- TYPE_YEARLY = 6: make yearly trigger
  - monthNumber[0 to 11] & dayNumber[1 to 31]
  - monthNumber[0 to 11] & dayType[0,1,2,3,4,5,6,7] & dayOrder[0,1,2,3,4]
- TYPE_CRON = 7: currently not supported

Parameters:

type -

---

getExecuting

```
public java.lang.Boolean getExecuting()
```

Gets the executing value for this ScheduleInfo.

Returns:

executing

---

setExecuting

```
public void setExecuting(java.lang.Boolean executing)
```

Sets the executing value for this ScheduleInfo.

Parameters:

executing -

---

getNextRuntime

```
public java.util.Calendar getNextRuntime()
```

Gets the nextRuntime value for this ScheduleInfo.

Returns:

nextRuntime

---

setNextRuntime

```
public void setNextRuntime(java.util.Calendar nextRuntime)
```

Sets the nextRuntime value for this ScheduleInfo.

Parameters:

```
nextRuntime -
```

---

getPrevRuntime

```
public java.util.Calendar getPrevRuntime()
```

Gets the prevRuntime value for this ScheduleInfo.

Returns:

```
prevRuntime
```

---

setPrevRuntime

```
public void setPrevRuntime(java.util.Calendar prevRuntime)
```

Sets the prevRuntime value for this ScheduleInfo.

Parameters:

```
prevRuntime -
```

---

getCronExpression

```
public java.lang.String getCronExpression()
```

Gets the cronExpression value for this ScheduleInfo.

Returns:

```
cronExpression
```

---

setCronExpression

```
public void setCronExpression(java.lang.String cronExpression)
```

Sets the schedule trigger info using quartz con expression string format, currently not supported. For example con expression “0 15 10 ? * *v” meaning “Fire at 10:15am every day”. read quartz documentation on how to create com expression.

Parameters:

`cronExpression -`

---

getDayNumber

```
public int getDayNumber()
```

Sets the dayNumber value for this ScheduleInfo. Sun=1,Mon=2,Tue=4,Wed=8,Thu=16,Fri=32, Sat=64. To set a combination such Sun and Mon set the dayNumber to 3 which was calculated by executing binary OR operation between Sun and Mon result 1 OR 2 = 3. dayNumber also could be used for schedule of type ”TYPE ”MONTHLY” and “TYPE_YEARLY”and then the valid of values dayNumber are from 1 to 31.

Returns:

`dayNumber -`

---

setDayNumber

```
public void setDayNumber(int dayNumber)
```

Sets the dayNumber value for this ScheduleInfo. Sun=1,Mon=2,Tue=4,Wed=8,Thu=16,Fri=32, Sat=64 to set a combination you should exec OR binary operation between desired days of week for sample a combination of Sun and Mon = 1 OR 2 in binary 01 OR 10 = 11 = in decimal 3. also it could be used for Type Monthly and then the values are 1-32.

Parameters:

`dayNumber -`

---

getMonthNumber

```
public int getMonthNumber()
```

Gets the monthNumber value for this ScheduleInfo.

Returns:

`monthNumber`

---

setMonthNumber

```
public void setMonthNumber(int monthNumber)
```

Sets the monthNumber value for this schedule Info. used for schedule of type ”TYPE _MONTHLY” and “TYPE_YEARLY”.Valid values are 0 – 11. sample "every day of the 3 of january".

Parameters:

```
monthNumber -
```

---

getDayType

```
public int getDayType()
```

Gets the dayType value for this ScheduleInfo.

Returns:

```
dayType
```

---

setDayType

```
public void setDayType(int dayType)
```

Sets the dayType value for this schedule Info. Used for schedule of type ” TYPE _MONTHLY” and “TYPE_YEARLY”. Valid values are:
Day = 0
SUNDAY = 1
MONDAY = 2
TUESDAY = 3
WEDNESDAY = 4
THURSDAY = 5
FRIDAY = 6
SATURDAY = 7

The dayType property requires the dayOrder property to be set and you will have to set dayNumber value to -1 if you want work with dayOrder and dayType.

Parameters:

```
dayType -
```

---

getDayOrder

```
public int getDayOrder()
```

Gets the dayOrder value for this ScheduleInfo.

Returns:

```
dayOrder
```

---

setDayOrder

```
public void setDayOrder(int dayOrder)
```

Sets the dayOrder value for this ScheduleInfo and used for type ”TYPE _MONTHLY” and “TYPE_ YEARLY”. Valid values are:
first = 0
second = 1
third = 2
fourth = 3
last = 4.

For example "first day of month" = "dayType=0 and dayOrder=0" or "first sunday of month" = "dayType=1 and dayOrder=0".

Parameters:

`dayOrder -`

---

getTriggerName

```
public java.lang.String getTriggerName()
```

Gets the trigger name value for this ScheduleInfo.

Returns:

`triggerName`

---

setTriggerName

```
public void setTriggerName(java.lang.String triggerName)
```

Gets the trigger name value for this ScheduleInfo.

Parameters:

`triggerName -`

---

getPaused

```
public java.lang.Boolean getPaused()
```

Return tells whether the schedule state is PAUSED or not.

Returns:

True if paused otherwise false.

---

setPaused

```
public void setPaused(java.lang.Boolean paused)
```

Not relevant as input.

Parameters:

paused -

ScheduleDisplayInfo

com.iconclude.dharma.scheduler.web

Class ScheduleDisplayInfo

java.lang.Object
|-com.iconclude.dharma.scheduler.web.ScheduleDisplayInfo extends ScheduleInfo

---

Serialized Fields

description

java.lang.String description

---

enabled

java.lang.Boolean enabled

---

endTime

java.util.Calendar endTime

---

name

java.lang.String name

---

params

Pair[] params

---

repeatCount

int repeatCount

---

repeatIntervalMilli

long repeatIntervalMilli

---

startTime

```
java.util.Calendar startTime
```

---

units

```
java.lang.String units
```

---

type

```
int type
```

---

executing

```
java.lang.Boolean executing
```

---

nextRuntime

```
java.util.Calendar nextRuntime
```

---

prevRuntime

```
java.util.Calendar prevRuntime
```

---

cronExpression

```
java.lang.String cronExpression
```

---

dayNumber

```
int dayNumber
```

---

monthNumber

```
int monthNumber
```

---

dayType

```
int dayType
```

---

dayOrder

```
int dayOrder
```

---

paused

```
java.lang.Boolean paused
```

---

triggerName

```
java.lang.String triggerName
```

Pair

```
com.iconclude.dharma.scheduler.web
```

```
Class Pair
```

```
java.lang.Object
               |-com.iconclude.dharma.scheduler.web.Pair
```

```
public class Pair
 extends java.lang.Object
```

This object is used to hold flow parameters as name and value pairs.

Constructor Summary

```
Pair()
```

```
Pair(java.lang.Object first, java.lang.Object second)
```

Methods Summary

Type	Method	Description
`java.lang.Object`	`getFirst()`	Gets the first value for this pair.
`java.lang.Object`	`getSecond()`	Gets the second value for this pair.
`void`	`setFirst(java.lang.Object first)`	Sets the first value for this pair.
`void`	`setSecond(java.lang.Object second)`	Sets the second value for this pair.

Constructor Detail

Pair

```
public Pair()
```

Pair

```
public Pair(java.lang.Object first,
            java.lang.Object second)
```

Methods Detail

getFirst

```
public java.lang.Object getFirst()
```

Gets the first value for this Pair.

Returns:

first

---

setFirst

```
public void setFirst(java.lang.Object first)
```

Sets the first value for this Pair.

Parameters:

```
first -
```

---

getSecond

```
public java.lang.Object getSecond()
```

Gets the second value for this Pair.

Returns:

second

---

setSecond

```
public void setSecond(java.lang.Object second)
```

Sets the Second value for this Pair.

Parameters:

```
second -
```

ScheduledFlowInfo

```
com.iconclude.dharma.scheduler.web

Class ScheduledFlowInfo

java.lang.Object
               |-com.iconclude.dharma.scheduler.web.ScheduledFlowInfo
```

```
public class ScheduledFlowInfo
 extends java.lang.Object
```

This object holds information about the current status of the scheduled flow and its scheduler.

Constructor Summary

```
ScheduledFlowInfo()

ScheduledFlowInfo(java.lang.String description, java.lang.Boolean
enabled,
java.lang.Boolean executing, java.lang.String lastRunReportURL,
java.lang.String lastRunReturnCode, java.lang.Boolean
lastRunSuccessful,
java.lang.String name, java.util.Calendar nextRuntime,
java.util.Calendar prevRuntime, java.lang.Boolean paused)
```

Methods Summary

The following table contains the method names, their type and a description of their effect on `ScheduledFlowInfo`.

Type	Method	Description
`java.lang.String`	`getDescription()`	Gets the description value.
`java.lang.Boolean`	`getEnabled()`	Gets the enabled value.
`java.lang.Boolean`	`getExecuting()`	Gets the executing value.
`java.lang.String`	`getFlowName()`	
`java.lang.String`	`getLastRunReportURL()`	Gets the `lastRunReportURL` .

Type	Method	Description
java.lang.String	getLastRunReturnCode()	Gets the lastRunReturnCode.
java.lang.Boolean	getLastRunSuccessful()	Gets the lastRunSuccessful.
java.lang.String	getName()	Gets the name.
java.util.Calendar	getNextRuntime()	Gets the nextRuntime.
java.util.List<java.lang.String>	getNextTriggerNames()	
java.lang.Boolean	getPaused()	
java.util.Calendar	getPrevRuntime()	Gets the prevRuntime.
java.util.List<java.lang.String>	getPrevTriggerNames()	
void	setDescription (java.lang.String description)	Sets the description value.
void	setEnabled (java.lang.Boolean enabled)	Sets the enabled value.
void	setExecuting (java.lang.Boolean executing)	Sets the executing value.
void	setFlowName (java.lang.String name)	
void	setLastRunReportURL (java.lang.String lastRunReportURL)	Sets the lastRunReportURL value.
void	setLastRunReturnCode (java.lang.String lastRunReturnCode)	Sets the lastRunReturnCode value.
void	setLastRunSuccessful (java.lang.Boolean lastRunSuccessful)	Sets the lastRunSuccessful value.

Type	Method	Description
void	setName(java.lang.String name)	Sets the name value.
void	setNextRuntime (java.util.Calendar nextRuntime)	Sets the nextRuntime value.
void	setNextTriggerNames (java.util.List<java.lang.String> nextTriggerNames)	
void	setPaused (java.lang.Boolean paused)	
void	setPrevRuntime (java.util.Calendar prevRuntime)	Sets the prevRuntime value.
void	setPrevTriggerNames (java.util.List<java.lang.String> prevTriggerNames)	

Constructor Detail

ScheduledFlowInfo

```
public ScheduledFlowInfo()
```

ScheduledFlowInfo

```
public ScheduledFlowInfo(java.lang.String description,
                java.lang.Boolean enabled,
                java.lang.Boolean executing,
                java.lang.String lastRunReportURL,
                java.lang.String lastRunReturnCode,
                java.lang.Boolean lastRunSuccessful,
                java.lang.String name,
                java.util.Calendar nextRuntime,
                java.util.Calendar prevRuntime,
                java.lang.Boolean paused)
```

Methods Detail

getDescription

```
public java.lang.String getDescription()
```

Gets the description value for this ScheduledFlowInfo.

Returns:

`description`

---

setDescription

```
public void setDescription(java.lang.String description)
```

Sets the description value for this `ScheduledFlowInfo`.

Parameters:

`description -`

---

getEnabled

```
public java.lang.Boolean getEnabled()
```

Gets the enabled value for this `ScheduledFlowInfo`.

Returns:

`enabled`

---

setEnabled

```
public void setEnabled(java.lang.Boolean enabled)
```

Sets the enabled value for this `ScheduledFlowInfo`.

Parameters:

`enabled`

---

getExecuting

```
public java.lang.Boolean getExecuting()
```

Gets the executing value for this `ScheduledFlowInfo`.

Returns:

`executing`

---

setExecuting

```
public void setExecuting(java.lang.Boolean executing)
```

Sets the executing value for this `ScheduledFlowInfo`.

Parameters:

`executing`

---

getLastRunReportURL

```
public java.lang.String getLastRunReportURL()
```

Gets the `lastRunReportURL` value for this `ScheduledFlowInfo`.

Returns:

`lastRunReportURL`

---

setLastRunReportURL

```
public void setLastRunReportURL(java.lang.String lastRunReportURL)
```

Sets the `lastRunReportURL` value for this `ScheduledFlowInfo`.

Parameters:

`lastRunReportURL -`

---

getLastRunReturnCode

```
public java.lang.String getLastRunReturnCode()
```

Gets the `lastRunReturnCode` value for this `ScheduledFlowInfo`.

Returns:

`lastRunReturnCode`

---

setLastRunReturnCode

```
public void setLastRunReturnCode(java.lang.String lastRunReturnCode)
```

Sets the `lastRunReturnCode` value for this `ScheduledFlowInfo`.

Parameters:

`lastRunReturnCode -`

---

getLastRunSuccessful

```
public java.lang.Boolean getLastRunSuccessful()
```

Gets the `lastRunSuccessful` value for this `ScheduledFlowInfo`.

Returns:

`lastRunSuccessful`

---

setLastRunSuccessful

```
public void setLastRunSuccessful(java.lang.Boolean lastRunSuccessful)
```

Sets the `lastRunSuccessful` value for this `ScheduledFlowInfo`.

Parameters:

`lastRunSuccessful -`

---

getName

```
public java.lang.String getName()
```

Gets the name value for this `ScheduledFlowInfo`.

Returns:

`name`

---

setName

```
public void setName(java.lang.String name)
```

Sets the name value for this `ScheduledFlowInfo`.

Parameters:

`name -`

---

getNextRuntime

```
public java.util.Calendar getNextRuntime()
```

Gets the `nextRuntime` value for this `ScheduledFlowInfo`.

Returns:

`nextRuntime`

---

setNextRuntime

`public void setNextRuntime(java.util.Calendar nextRuntime)`

Sets the `nextRuntime` value for this `ScheduledFlowInfo`.

Parameters:

`nextRuntime -`

---

getPrevRuntime

`public java.util.Calendar getPrevRuntime()`

Gets the `prevRuntime` value for this `ScheduledFlowInfo`.

Returns:

`prevRuntime`

---

setPrevRuntime

`public void setPrevRuntime(java.util.Calendar prevRuntime)`

Sets the `prevRuntime` value for this `ScheduledFlowInfo`.

Parameters:

`prevRuntime -`

---

getFlowName

`public java.lang.String getFlowName()`

Returns the name of the flow that was scheduled.

---

setFlowName

`public void setFlowName(java.lang.String name)`

Sets the name of the flow being scheduled.

---

getNextTriggerNames

```
public java.util.List<java.lang.String> getNextTriggerNames()
```

Returns a list containing the names of the schedules being executed next.

---

setNextTriggerNames

```
public void setNextTriggerNames(java.util.List<java.lang.String>
nextTriggerNames)
```

Sets the list containing the names of the schedules being executed next, ordered chronologically.

---

getPrevTriggerNames

```
public java.util.List<java.lang.String> getPrevTriggerNames()
```

Returns a list containing the names of the previously triggered schedules.

---

setPrevTriggerNames

```
public void setPrevTriggerNames(java.util.List<java.lang.String>
prevTriggerNames)
```

Sets the list containing the names of the previously triggered schedules.

---

getPaused

```
public java.lang.Boolean getPaused()
```

Returns `true`, if all the schedules of this flow are paused. Otherwise, returns `false`.

---

setPaused

```
public void setPaused(java.lang.Boolean paused)
```

If `true`, sets all the flow schedules to `paused`. Otherwise, all the schedules are `executing`.

Methods for working with Scheduler

scheduleFlow

This method creates a new schedule for a flow:

```
scheduleFlow(String flowUuid, ScheduleInfo info)
```

```
ScheduleInfo {
       String description
       Boolean enabled
       Calendar endTime
       String name
       Pair[] params
       int repeatCount
       long repeatIntervalMilli
       Calendar startTime
       String units
       int type
       Boolean executing
       Calendar nextRuntime
       Calendar prevRuntime
       String cronExpression
       int dayNumber
       int monthNumber
       int dayType
       int dayOrder
       Boolean paused

}
Pair {
       Object first
       Object second
}
```

- `description` is the description of the schedule.
- `enabled` determines whether the schedule is enabled. The schedule is considered enabled when its state is:
  - NORMAL
  - BLOCKED
- `endTime` is the end time of the schedule. The valid pattern is: `YYYY-MM-DDTHH:MM:SS`
- `name` is the name of the schedule, in the form of UUID. If the provided name already exists in the database, the corresponding schedule will be updated.
- `params` is the set of inputs.
- `repeatCount` is the number of schedule occurrences, if the schedule type is `TYPE_INTERVAL`.
- `repeatIntervalMilli` is the interval to repeat the schedule (in milliseconds), if the schedule is `TYPE_INTERVAL`.
- `startTime` is the start time of the schedule. The valid pattern is: `YYYY-MM-DDTHH:MM:SS`
- `units` represents the unit type of the recurrence.

- `type` is the type of the schedule. `type` can have the following values:
  - 1 = `TYPE_INTERVAL`
  - 2 = `TYPE_EVERY_DAY`
  - 3 = `TYPE_EVERY_WEEKDAY`
  - 4 = `TYPE_WEEKLY`
  - 5 = `TYPE_MONTHLY`
  - 6 = `TYPE_YEARLY`
- `executing` determines whether the schedule is executing or not.
- `nextRuntime` is the date of the next run. The valid pattern is: `YYYY-MM-DDTHH:MM:SS`
- `prevRuntime` is the date of the previous run. The valid pattern is `YYYY-MM-DDTHH:MM:SS`
- `cronExpression` is the string representing the `quartz cron` expression that defines the pattern of the schedule.
- `dayNumber` represents the number of the day in which the schedule runs. This is available only for:
  - `TYPE_WEEKLY`
    - 1 = Sunday
    - 2 = Monday
    - 4 = Tuesday
    - 8 = Wednesday
    - 16 = Thursday
    - 32 = Friday
    - 64 = Saturday
  - `TYPE_MONTHLY` and `TYPE_YEARLY`
    - Numbers between 1 and 31
- `monthNumber` represents the monthly recurrence.
- `dayType` represents the type of the day for the schedule. This is available for the `TYPE_MONTHLY` and `TYPE_YEARLY` schedules:
  - 0 = day
  - 1 = Sunday
  - 2 = Monday
  - 3 = Tuesday
  - 4 = Wednesday
  - 5 = Thursday
  - 6 = Friday
  - 7 = Saturday

- `dayOrder` represents the order of the day. This is available only for the `TYPE_MONTHLY` schedule:
  - 0 = first
  - 1 = second
  - 2 = third
  - 3 = fourth
  - 4 = last
- `paused` determines whether the schedule is paused.

Inputs

Input	Type	Description
flowUuids	String	The UUID of the flow to be scheduled.
info	ScheduleInfo	The schedule information.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:scheduleFlow soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuid xsi:type="soapenc:string">3541d63f-603a-449b-9d43-8e57d7d61482</flowUuid>
         <info xsi:type="soap:ScheduleInfo"
xmlns:soap="http://iconclude.com/webservices/rss/v2.0/soap">
            <cronExpression xsi:type="xsd:string"></cronExpression>
            <dayNumber xsi:type="xsd:int">17</dayNumber>
            <dayOrder xsi:type="xsd:int">0</dayOrder>
            <dayType xsi:type="xsd:int">0</dayType>
            <description xsi:type="xsd:string">this is a schedule</description>
            <enabled xsi:type="xsd:boolean">true</enabled>
            <endTime xsi:type="xsd:dateTime">2012-12-29T18:30:00+02:00</endTime>
            <executing xsi:type="xsd:boolean">true</executing>
            <monthNumber xsi:type="xsd:int">2</monthNumber>
            <name xsi:type="xsd:string">3541d63f-603a-449b-9d43-8e57d7d61482</name>
            <nextRuntime xsi:type="xsd:dateTime">2011-12-29T18:30:00+02:00</nextRuntime>
            <params xsi:type="wsc:ArrayOf_tns2_Pair" soapenc:arrayType="soap:Pair[]"
xmlns:wsc="https://my-ooserver.myco.com:8443/PAS/services/WSCentralService">
          </params>
            <paused xsi:type="xsd:boolean">false</paused>
            <prevRuntime xsi:type="xsd:dateTime">2011-12-29T18:30:00+02:00</prevRuntime>
            <repeatCount xsi:type="xsd:int">0</repeatCount>
```

```
            <repeatIntervalMilli xsi:type="xsd:long">0</repeatIntervalMilli>
            <startTime xsi:type="xsd:dateTime">2011-12-29T18:30:00+02:00</startTime>
            <triggerName xsi:type="xsd:string">3541d63f-603a-449b-9d43-
8e57d7d61482</triggerName>
            <type xsi:type="xsd:int">5</type>
            <units xsi:type="xsd:string">0</units>
         </info>
      </wsc:scheduleFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:scheduleFlowResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

getSchedule

This method retrieves information about a specific schedule:

```
ScheduleInfo getSchedule(String schedule)
```

```
ScheduleInfo {
       String description
       Boolean enabled
       Calendar endTime
       String name
       Pair[] params
       int repeatCount
       long repeatIntervalMilli
       Calendar startTime
       String units
       int type
       Boolean executing
       Calendar nextRuntime
       Calendar prevRuntime
       String cronExpression
       int dayNumber
       int monthNumber
       int dayType
       int dayOrder
       Boolean paused

}
Pair {
       Object first
       Object second
}
```

- `description` is the description of the schedule.
- `enabled` determines whether the schedule is enabled. The schedule is considered enabled when its state is:

  - NORMAL
  - BLOCKED
- `endTime` is the end time of the schedule. The valid pattern is: `YYYY-MM-DDTHH:MM:SS`
- `name` is the name of the schedule.
- `params` is the set of inputs.
- `repeatCount` is the number of schedule occurrences, if the schedule type is `TYPE_INTERVAL`.
- `repeatIntervalMilli` is the interval to repeat the schedule (in milliseconds), if the schedule is `TYPE_INTERVAL`.
- `startTime` is the start time of the schedule. The valid pattern is: `YYYY-MM-DDTHH:MM:SS`
- `units` represents the unit type of the recurrence.
- `type` is the type of the schedule. `type` can have the following values:
  - 1 = `TYPE_INTERVAL`
  - 2 = `TYPE_EVERY_DAY`
  - 3 = `TYPE_EVERY_WEEKDAY`
  - 4 = `TYPE_WEEKLY`
  - 5 = `TYPE_MONTHLY`
  - 6 = `TYPE_YEARLY`
- `executing` determines whether the schedule is executing or not.
- `nextRuntime` is the date of the next run. The valid pattern is: `YYYY-MM-DDTHH:MM:SS`
- `prevRuntime` is the date of the previous run. The valid pattern is `YYYY-MM-DDTHH:MM:SS`
- `cronExpression` is the string representing the `quartz cron` expression that defines the pattern of the schedule.
- `dayNumber` represents the number of the day in which the schedule runs. This is available only for:
  - `TYPE_WEEKLY`
    - 1 = Sunday
    - 2 = Monday
    - 4 = Tuesday
    - 8 = Wednesday
    - 16 = Thursday
    - 32 = Friday
    - 64 = Saturday
  - `TYPE_MONTHLY` and `TYPE_YEARLY`
    - Numbers between 1 and 31
- `monthNumber` represents the monthly recurrence.

- `dayType` represents the type of the day for the schedule. This is available for the `TYPE_MONTHLY` and `TYPE_YEARLY` schedules:
  - 0 = day
  - 1 = Sunday
  - 2 = Monday
  - 3 = Tuesday
  - 4 = Wednesday
  - 5 = Thursday
  - 6 = Friday
  - 7 = Saturday
- `dayOrder` represents the order of the day. This is available only for the `TYPE_MONTHLY` schedule:
  - 0 = first
  - 1 = second
  - 2 = third
  - 3 = fourth
  - 4 = last
- `paused` determines whether the schedule is paused.

Inputs

Input	Type	Description
schedule	String	The unique identifier of the schedule (UUID).

Outputs

Output	Type	Description
Result	ScheduleInfo	The schedule information.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
```

```
   <soapenv:Body>
      <wsc:getSchedule soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <schedule xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">d7101e7f-e154-4ff5-888b-
dbcf4abe4904</schedule>
      </wsc:getSchedule>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getScheduleResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getScheduleReturn xsi:type="ns2:ScheduleInfo"
xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap">
            <cronExpression xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">0 0 17 1 1 ? 2013</cronExpression>
            <dayNumber xsi:type="xsd:int">-1</dayNumber>
            <dayOrder xsi:type="xsd:int">0</dayOrder>
            <dayType xsi:type="xsd:int">0</dayType>
            <description xsi:type="soapenc:string" xsi:nil="true"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
            <enabled xsi:type="soapenc:boolean"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">true</enabled>
            <endTime xsi:type="xsd:dateTime" xsi:nil="true"/>
            <executing xsi:type="soapenc:boolean"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">false</executing>
            <monthNumber xsi:type="xsd:int">4</monthNumber>
            <name xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">d7101e7f-e154-4ff5-888b-
dbcf4abe4904</name>
            <nextRuntime xsi:type="xsd:dateTime">2013-01-01T22:00:00.000Z</nextRuntime>
            <params soapenc:arrayType="ns2:Pair[1]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
               <params xsi:type="ns2:Pair">
                  <first xsi:type="soapenc:string">host</first>
                  <second xsi:type="soapenc:string">localhsot</second>
               </params>
            </params>
            <paused xsi:type="soapenc:boolean"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">false</paused>
            <prevRuntime xsi:type="xsd:dateTime" xsi:nil="true"/>
            <repeatCount xsi:type="xsd:int">-1</repeatCount>
            <repeatIntervalMilli xsi:type="xsd:long">0</repeatIntervalMilli>
            <startTime xsi:type="xsd:dateTime">2012-09-30T21:00:00.000Z</startTime>
            <triggerName xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Windows Health Check</triggerName>
            <type xsi:type="xsd:int">5</type>
            <units xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">millisec</units>
         </getScheduleReturn>
      </ns1:getScheduleResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

pauseSchedule

This method allows you to pause a specific schedule:

```
pauseSchedule(String schedule)
```

Inputs

Input	Type	Description
schedule	String	The unique identifier of the schedule (UUID).

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:pauseSchedule soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <schedule xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">d7101e7f-e154-4ff5-888b-
dbcf4abe4904</schedule>
      </wsc:pauseSchedule>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:pauseScheduleResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

isSchedulePaused

This method returns `true` if a scheduled job is paused. Otherwise, returns `false`.

```
boolean isSchedulePaused(String schedule)
```

Inputs

Input	Type	Description
schedule	String	The unique identifier of the schedule (UUID).

Outputs

Output	Type	Description
Result	Boolean	`true`, if the schedule is paused. Otherwise, returns `false`.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:isSchedulePaused soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <schedule xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">d7101e7f-e154-4ff5-888b-
dbcf4abe4904</schedule>
      </wsc:isSchedulePaused>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:isSchedulePausedResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <isSchedulePausedReturn xsi:type="xsd:boolean">false</isSchedulePausedReturn>
      </ns1:isSchedulePausedResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

resumeSchedule

This method resumes a scheduled job, identified by the UUID:

`resumeSchedule(String schedule)`

Inputs

Input	Type	Description
schedule	String	The unique identifier of the schedule (UUID).

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:resumeSchedule soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <schedule xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">d7101e7f-e154-4ff5-888b-
dbcf4abe4904</schedule>
      </wsc:resumeSchedule>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:resumeScheduleResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

deleteSchedule

This method deletes a scheduled job, identified by the UUID:

`deleteSchedule(String schedule)`

Inputs

Input	Type	Description
schedule	String	The unique identifier of the schedule (UUID).

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:deleteSchedule soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <schedule xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">ad4cc6fc-042d-471a-9282-
e5ec034baa59</schedule>
      </wsc:deleteSchedule>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:deleteScheduleResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

isSchedulerEnabled

This method returns `true` if the **Scheduler** service is enabled. Otherwise, returns `false`.

```
Boolean isSchedulerEnabled()
```

Outputs

Output	Type	Description
Result	Boolean	`true`, if the scheduler is enabled. Otherwise, returns `false`.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:isSchedulerEnabled
```

```
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:isSchedulerEnabledResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <isSchedulerEnabledReturn xsi:type="xsd:boolean">true</isSchedulerEnabledReturn>
      </ns1:isSchedulerEnabledResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getScheduledFlows

This method returns a list containing all the scheduled flows:

```
ScheduledFlowInfo[] getScheduledFlows()
```

```
ScheduledFlowInfo {
       String description;
       Boolean enabled;
       Boolean executing;
       String lastRunReportURL;
       String lastRunReturnCode;
       Boolean lastRunSuccessful;
       String name;
       Calendar nextRuntime;
       Calendar prevRuntime;
       String flowName;
       String[] nextTriggerNames;
       String[] prevTriggerNames;
       Boolean paused;

}
```

- `description` is the description set to the flow.
- `enabled` determines whether the schedule is enabled.
- `executing` determines whether the schedule is executing.
- `lastRunReportURL` is the URL of the last schedule run's report.
- `lastRunReturnCode` is the return code of the last schedule run.
- `lastRunSuccessful` determines whether the last run of the schedule was a success or failure.
- `name` is the name given to the schedule.
- `nextRunTime` is the date of the next run time.
- `prevRunTime` is the date of the previous run time.
- `flowName` is the name of the flow that was scheduled.

- `nextTriggerNames` is a list containing the names of the schedules to be triggered from the flow.
- `prevTriggerNames` is a list containing the names of the schedules previously triggered from the flow.
- `paused` determines whether or not the the schedule is paused.

Outputs

Output	Type	Description
Result	ScheduledFlowInfo[]	An array containing the schedule information.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getScheduledFlows
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getScheduledFlowsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getScheduledFlowsReturn soapenc:arrayType="ns2:ScheduleFlowInfo[2]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getScheduledFlowsReturn xsi:type="ns2:ScheduleFlowInfo">
               <description xsi:type="soapenc:string" xsi:nil="true"/>
               <enabled xsi:type="soapenc:boolean">true</enabled>
               <executing xsi:type="soapenc:boolean">false</executing>
               <flowName xsi:type="soapenc:string" xsi:nil="true"/>
               <lastRunReportURL xsi:type="soapenc:string" xsi:nil="true"/>
               <lastRunReturnCode xsi:type="soapenc:string"/>
               <lastRunSuccessful xsi:type="soapenc:boolean" xsi:nil="true"/>
               <name xsi:type="soapenc:string">7107fd7d-5964-4016-a4a8-9f412df96c7e</name>
               <nextRuntime xsi:type="xsd:dateTime">2012-11-01T21:00:00.000Z</nextRuntime>
               <nextTriggerNames soapenc:arrayType="xsd:anyType[1]" xsi:type="soapenc:Array">
                  <nextTriggerNames xsi:type="soapenc:string">test_flow</nextTriggerNames>
               </nextTriggerNames>
```

```
               <paused xsi:type="soapenc:boolean">false</paused>
               <prevRuntime xsi:type="xsd:dateTime" xsi:nil="true"/>
               <prevTriggerNames xsi:type="soapenc:Array" xsi:nil="true"/>
            </getScheduledFlowsReturn>
            <getScheduledFlowsReturn xsi:type="ns2:ScheduleFlowInfo">
               <description xsi:type="soapenc:string" xsi:nil="true"/>
               <enabled xsi:type="soapenc:boolean">true</enabled>
               <executing xsi:type="soapenc:boolean">false</executing>
               <flowName xsi:type="soapenc:string" xsi:nil="true"/>
               <lastRunReportURL xsi:type="soapenc:string" xsi:nil="true"/>
               <lastRunReturnCode xsi:type="soapenc:string"/>
               <lastRunSuccessful xsi:type="soapenc:boolean" xsi:nil="true"/>
               <name xsi:type="soapenc:string">d012e1c3-704f-426f-a380-b2425a166d39</name>
               <nextRuntime xsi:type="xsd:dateTime">2013-06-28T13:52:00.000Z</nextRuntime>
               <nextTriggerNames soapenc:arrayType="xsd:anyType[1]" xsi:type="soapenc:Array">
                  <nextTriggerNames xsi:type="soapenc:string">How do I: Create a parallel
flow</nextTriggerNames>
               </nextTriggerNames>
               <paused xsi:type="soapenc:boolean">false</paused>
               <prevRuntime xsi:type="xsd:dateTime">2013-03-28T13:52:00.000Z</prevRuntime>
               <prevTriggerNames soapenc:arrayType="xsd:anyType[1]" xsi:type="soapenc:Array">
                  <prevTriggerNames xsi:type="soapenc:string">How do I: Create a parallel
flow</prevTriggerNames>
               </prevTriggerNames>
            </getScheduledFlowsReturn>
         </getScheduledFlowsReturn>
      </ns1:getScheduledFlowsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getSchedulesOfFlow

This method retrieves the details of a flow's schedules:

```
WSScheduleDetails[] getSchedulesOfFlow(String flowUuid)
```

```
WSScheduleDetailes {
       String name
       String description
       String cronExpression
       String flowUuid
       Boolean executing
       Boolean enabled
       Long repeatIntervalMilli
       Long endTime
       Long startTime
       Long nextRuntime
       Long prevRuntime
       WSScheduleUnitType unit
       WSScheduleType type
       String[][] inputs

}
WSScheduleUnitType {
       int unit
       String description
}
WSScheduleType {
       int type
```

```
       String description
}
```

Inputs

Input	Type	Description
flowUuid	String	The UUID of the flow for which to retrieve the schedules.

Outputs

Output	Type	Description
Result	WSScheduleDetails[]	An array containing the details of each schedule.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getSchedulesOfFlow
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuid xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">2c54f8f3-a8f0-4634-9aec-
7d479d9c7321</flowUuid>
      </wsc:getSchedulesOfFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getSchedulesOfFlowResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getSchedulesOfFlowReturn soapenc:arrayType="ns2:WSScheduleDetails[2]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getSchedulesOfFlowReturn xsi:type="ns2:WSScheduleDetails">
               <cronExpression xsi:type="soapenc:string" xsi:nil="true"/>
               <description xsi:type="soapenc:string" xsi:nil="true"/>
               <enabled xsi:type="soapenc:boolean">true</enabled>
               <endTime xsi:type="xsd:long">0</endTime>
               <executing xsi:type="soapenc:boolean">false</executing>
               <flowUuid xsi:type="soapenc:string">2c54f8f3-a8f0-4634-9aec-
```

```
7d479d9c7321</flowUuid>
               <name xsi:type="soapenc:string">8ad7c3c5-d51d-475e-a3fc-3b5bb03d8b11</name>
               <nextRuntime xsi:type="xsd:long">1348753500000</nextRuntime>
               <params soapenc:arrayType="soapenc:string[][1]" xsi:type="soapenc:Array">
                  <params soapenc:arrayType="soapenc:string[2]" xsi:type="soapenc:Array">
                     <params xsi:type="soapenc:string">host</params>
                     <params xsi:type="soapenc:string">my-ooserver.myco.com</params>
                  </params>
               </params>
               <prevRuntime xsi:type="xsd:long">1348753200000</prevRuntime>
               <repeatIntervalMilli xsi:type="xsd:long">300000</repeatIntervalMilli>
               <startTime xsi:type="xsd:long">1328562000000</startTime>
               <type xsi:type="ns2:WSScheduleType">
                  <description xsi:type="soapenc:string">TYPE_INTERVAL</description>
                  <type xsi:type="xsd:int">1</type>
               </type>
               <unit xsi:type="ns2:WSScheduleUnitType">
                  <description xsi:type="soapenc:string">UNIT_MINUTES</description>
                  <unit xsi:type="xsd:int">1</unit>
               </unit>
            </getSchedulesOfFlowReturn>
            <getSchedulesOfFlowReturn xsi:type="ns2:WSScheduleDetails">
               <cronExpression xsi:type="soapenc:string" xsi:nil="true"/>
               <description xsi:type="soapenc:string" xsi:nil="true"/>
               <enabled xsi:type="soapenc:boolean">true</enabled>
               <endTime xsi:type="xsd:long">0</endTime>
               <executing xsi:type="soapenc:boolean">false</executing>
               <flowUuid xsi:type="soapenc:string">2c54f8f3-a8f0-4634-9aec-
7d479d9c7321</flowUuid>
               <name xsi:type="soapenc:string">9b4666aa-c200-4317-aaa8-18c7a0da4592</name>
               <nextRuntime xsi:type="xsd:long">1348753500000</nextRuntime>
               <params soapenc:arrayType="soapenc:string[][1]" xsi:type="soapenc:Array">
                  <params soapenc:arrayType="soapenc:string[2]" xsi:type="soapenc:Array">
                     <params xsi:type="soapenc:string">host</params>
                     <params xsi:type="soapenc:string">localhost</params>
                  </params>
               </params>
               <prevRuntime xsi:type="xsd:long">1348753200000</prevRuntime>
               <repeatIntervalMilli xsi:type="xsd:long">300000</repeatIntervalMilli>
               <startTime xsi:type="xsd:long">1328562000000</startTime>
               <type xsi:type="ns2:WSScheduleType">
                  <description xsi:type="soapenc:string">TYPE_INTERVAL</description>
                  <type xsi:type="xsd:int">1</type>
               </type>
               <unit xsi:type="ns2:WSScheduleUnitType">
                  <description xsi:type="soapenc:string">UNIT_MINUTES</description>
                  <unit xsi:type="xsd:int">1</unit>
               </unit>
            </getSchedulesOfFlowReturn>
         </getSchedulesOfFlowReturn>
      </ns1:getSchedulesOfFlowResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

getSchedulesForFlowCategory

This method retrieves the schedules the of flows belonging to a certain category:

```
WSScheduleDetails[] getSchedulesForFlowCategory(String flowCategory)
```

```
WSScheduleDetailes {
       String name
       String description
       String cronExpression
       String flowUuid
       Boolean executing
       Boolean enabled
       Long repeatIntervalMilli
       Long endTime
       Long startTime
       Long nextRuntime
       Long prevRuntime
       WSScheduleUnitType unit
       WSScheduleType type
       String[][] inputs

}
WSScheduleUnitType {
       int unit
       String description
}
WSScheduleType {
       int type
       String description
}
```

Inputs

Input	Type	Description
flowCategory	String	The name of the category used as search criteria.

Outputs

Output	Type	Description

Result	WSScheduleDetails []	An array containing the details of every schedule, including: • The name of the schedule. • The description of the schedule. • The `quartz cron` expression of the schedule. • The flow UUID. • Whether the schedule is running. • Whether the schedule is enabled. • The repeat interval (in milliseconds), if the schedule type is `INTERVAL`. • The end time (in milliseconds) since 1/1/1970. • The start time (in milliseconds) since 1/1/1970. • The previous runtime (in milliseconds), since 1/1/1970. • The next runtime (in milliseconds), since 1/1/1970. • The schedule recurrence's time unit. • The schedule type. • The schedule inputs.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:getSchedulesForFlowCategory
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowCategory xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Network</flowCategory>
      </wsc:getSchedulesForFlowCategory>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:getSchedulesForFlowCategoryResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <getSchedulesForFlowCategoryReturn soapenc:arrayType="ns2:WSScheduleDetails[1]"
xsi:type="soapenc:Array" xmlns:ns2="http://iconclude.com/webservices/rss/v2.0/soap"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <getSchedulesForFlowCategoryReturn xsi:type="ns2:WSScheduleDetails">
               <cronExpression xsi:type="soapenc:string">0 0 17 1 11 ?</cronExpression>
               <description xsi:type="soapenc:string" xsi:nil="true"/>
               <enabled xsi:type="soapenc:boolean">true</enabled>
               <endTime xsi:type="xsd:long">0</endTime>
               <executing xsi:type="soapenc:boolean">false</executing>
               <flowUuid xsi:type="soapenc:string">7107fd7d-5964-4016-a4a8-
9f412df96c7e</flowUuid>
               <name xsi:type="soapenc:string">acf5b435-57d8-4dd5-8dab-39534edd0c6c</name>
               <nextRuntime xsi:type="xsd:long">1351803600000</nextRuntime>
               <params soapenc:arrayType="soapenc:string[][0]" xsi:type="soapenc:Array"/>
               <prevRuntime xsi:type="xsd:long">0</prevRuntime>
               <repeatIntervalMilli xsi:type="xsd:long">0</repeatIntervalMilli>
               <startTime xsi:type="xsd:long">1349038800000</startTime>
               <type xsi:type="ns2:WSScheduleType">
                  <description xsi:type="soapenc:string">TYPE_CRON</description>
                  <type xsi:type="xsd:int">2</type>
               </type>
               <unit xsi:type="ns2:WSScheduleUnitType">
                  <description xsi:type="soapenc:string">UNIT_UNKNOWN</description>
                  <unit xsi:type="xsd:int">0</unit>
               </unit>
            </getSchedulesForFlowCategoryReturn>
         </getSchedulesForFlowCategoryReturn>
      </ns1:getSchedulesForFlowCategoryResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

pauseScheduledFlow

This method pauses the schedules of a flow:

`pauseScheduledFlow(String flowUuid)`

Inputs

Input	Type	Description
flowUuid	String	The UUID of the flow whose schedules are to be paused.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:pauseScheduledFlow
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuid xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">3541d63f-603a-449b-9d43-
8e57d7d61482</flowUuid>
      </wsc:pauseScheduledFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:pauseScheduledFlowResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

isScheduledFlowPaused

This method returns whether a flow's schedules are paused or not:

`Boolean isScheduledFlowPaused(String flowUuid)`

Inputs

Input	Type	Description
flowUuid	String	The flow UUID.

Outputs

Output	Type	Description
Result	Boolean	Returns `true` if all the schedules of the flow are paused. Otherwise, returns `false`.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:isScheduledFlowPaused
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuid xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">2c54f8f3-a8f0-4634-9aec-
7d479d9c7321</flowUuid>
      </wsc:isScheduledFlowPaused>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:isScheduledFlowPausedResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com">
         <isScheduledFlowPausedReturn
xsi:type="xsd:boolean">false</isScheduledFlowPausedReturn>
      </ns1:isScheduledFlowPausedResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

resumeScheduledFlow

This method resumes the schedules of a flow:

`resumeScheduledFlow(String flowUuid)`

Inputs

Input	Type	Description
flowUuid	String	The UUID of the flow.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:resumeScheduledFlow
```

```
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuid xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">3541d63f-603a-449b-9d43-
8e57d7d61482</flowUuid>
      </wsc:resumeScheduledFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:resumeScheduledFlowResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

deleteScheduledFlow

This methods enables you to delete all the schedules of a flow:

`deleteScheduledFlow(String flowUuid)`

Inputs

Input	Type	Description
flowUuid	String	The UUID of the flow.

Exceptions

`AxisFault` is thrown if:

- There is an execution error.
- There is a violation error.

Example

- **Request**:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsc="http://wscentralservice.services.dharma.iconclude.com">
   <soapenv:Header/>
   <soapenv:Body>
      <wsc:deleteScheduledFlow
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <flowUuid xsi:type="soapenc:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">7107fd7d-5964-4016-a4a8-
9f412df96c7e</flowUuid>
      </wsc:deleteScheduledFlow>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Response**:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soapenv:Body>
      <ns1:deleteScheduledFlowResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://wscentralservice.services.dharma.iconclude.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

Selection lists

The methods documented in this section enable you to handle selection lists:

- "createSelectionList" below.
- "getSelectionList" below.

createSelectionList

This method creates or updates a `Selection List` object:

```
String createSelectionList(String name, String description, String[] values)
```

The `Selection List` objects are located in the **Configuration/Selection Lists** folder. This method creates a selection list using the `name`, `description`, and `values` populated from the inputs under the **Configuration/Selection Lists** repository path . If the list already exists, it is updated with the new values (`description` and `values`).

Inputs

Input	Type	Description
name	String	The name of the selection list.
description	String	The description of the selection list.
values	String[]	The values of the selection list.

Outputs

Output	Type	Description
Result	String	The UUID of the created or updated `Selection List` object.

Exceptions

Exceptions are thrown if:

- The `name` input is empty or if it contains illegal characters.

  The legal characters are alphanumeric, spaces or the following: `_, __, -, ¯, :, :, ., ·, ., ◦, ◦, ▪, ·, (, ), (, ), [, ], [, ], {, }, {, }`
- The user does not have the necessary permissions.
- The `values` input:
  - Is null or contains no elements.
  - Contains null elements.
  - Contains duplicate elements.

getSelectionList

This method gets information about a `Selection List` object:

```
WSListDetails getSelectionList(String name)

WSListDetails {
       String uuid;
       String name;
       String description;
       String[] values;
       String version;
       String comment;
       String lastModifiedBy;
       String lastModified;

}
```

The `Selection List` objects are located in the **Configuration/Selection Lists** folder. The method returns information about the specified selection list, including the following:

Element	Description
`UUID`	The selection list's universally unique identifier.
`name`	The name of the selection list.
`description`	The description of the selection list.
`values`	An array containing the selection list's values.
`version`	The version number of the selection list.
`comment`	A comment.
`lastModifiedBy`	The last person to modify the selection list.
`lastModified`	The last date when selection list was modified.

Inputs

Input	Type	Description
name	String	The name or the UUID of the selection list.

Outputs

Output	Type	Description
Result	WSListDetails	A `WSListDetails` object containing information about the selected `Selection List` object.

Exceptions

Exceptions are thrown if the selection list does not exist. For example, a selection list having the name or the UUID equal to the `name` input cannot be found in the **Configuration/Selection Lists** folder.


Java™
COMPATIBLE