CA Workload Automation Agent for Informatica

CLI User Guide
r11.3.1
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

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

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

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

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

The manufacturer of this Documentation is CA.

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

Copyright © 2013 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
CA Technologies Product References

This document references the following CA Technologies products:

- CA Process Automation
- CA Workload Automation AE
- CA Workload Automation Agent for Application Services (CA WA Agent for Application Services)
- CA Workload Automation Agent for Databases (CA WA Agent for Databases)
- CA Workload Automation Agent for i5/OS (CA WA Agent for i5/OS)
- CA Workload Automation Agent for Informatica (CA WA Agent for Informatica)
- CA Workload Automation Agent for Linux (CA WA Agent for Linux)
- CA Workload Automation Agent for Micro Focus (CA WA Agent for Micro Focus)
- CA Workload Automation Agent for Microsoft SQL Server (CA WA Agent for Microsoft SQL Server)
- CA Workload Automation Agent for Oracle E-Business Suite (CA WA Agent for Oracle E-Business Suite)
- CA Workload Automation Agent for PeopleSoft (CA WA Agent for PeopleSoft)
- CA Workload Automation Agent for Remote Execution (CA WA Agent for Remote Execution)
- CA Workload Automation Agent for SAP (CA WA Agent for SAP)
- CA Workload Automation Agent for UNIX (CA WA Agent for UNIX)
- CA Workload Automation Agent for Web Services (CA WA Agent for Web Services)
- CA Workload Automation Agent for Windows (CA WA Agent for Windows)
- CA Workload Automation CA 7 Edition
- CA Workload Automation DE
- CA Workload Automation Desktop Client (CA WA Desktop Client)
- CA Workload Automation ESP Edition
- CA Workload Control Center
Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.
## Contents

### Chapter 1: Overview

- About This Guide .................................................................................................................. 7
- Using the Command Line Interface ....................................................................................... 7
- Memory Limitations ................................................................................................................. 9
- Workflow Run ID Not Supported ............................................................................................ 9

### Chapter 2: Integrating with Informatica PowerCenter

- Informatica Command Line Interface .................................................................................. 11
- Invoking the Informatica CLI ............................................................................................... 12
- informatica Script—Integrate with Informatica PowerCenter .................................................. 13
  - Example: Start a Workflow on Informatica from a Task ...................................................... 24
  - Example: Start a Workflow on Informatica from the Beginning ........................................ 24
  - Example: Start a Workflow on Informatica with a Parameter File ................................... 25
  - Example: List Workflows that Match a Specified Pattern ................................................ 25
  - Example: List All Workflows in a Folder .......................................................................... 26
  - Example: Display All Tasks for a Workflow ...................................................................... 27
  - Example: Display the Sequence of Tasks in a Workflow .................................................. 27
  - Example: Display the Sequence of the Tasks in a Workflow Using Verbose .................... 28
  - Example: Display the Hierarchy of a Task Within a Workflow .......................................... 29
  - Example: Display the Hierarchy of a Workflow Using Verbose ......................................... 30
  - Example: Display the Details of a Task Within a Workflow ............................................. 31
  - Example: Retrieve the Status of a Workflow Instance Using Verbose ............................ 31
  - Example: Retrieve the Log of a Workflow Instance Using Verbose .................................. 32
  - Example: Display the Run Details of a Workflow ............................................................... 34
  - Example: Monitor a Workflow Using Verbose .................................................................. 36
  - Example: Restart a Workflow Instance Without Tracking ................................................. 37
  - Example: Restart a Workflow Using Tracking ................................................................. 38
  - Example: Stop a Running Workflow Instance ................................................................... 39
- Exit Codes for Job Submission .............................................................................................. 39
- Exit Codes for Task Failures ................................................................................................. 40

### Chapter 3: Running the CLI Using Your Scheduling Manager

- Integration with CA Workload Automation DE .................................................................. 47
  - Example: List the Workflows that Match a Specified Pattern Using a Windows Job .......... 47
  - Example: Start a Workflow Using a UNIX Job ................................................................. 48
  - Example: Stop a Running Workflow Using a Windows Job ............................................ 49
Integration with CA Workload Automation AE ................................................................. 49
  Example: List the Workflows that Match a Specified Pattern Using a Command Job ............. 50
  Example: Start a Workflow Using a Command Job ............................................................ 51
  Example: Stop a Running Workflow Using a Command Job ............................................... 51
Integration with CA Workload Automation ESP Edition ..................................................... 52
  Example: List the Workflows that Match a Specified Pattern Using a Windows Job .................. 52
  Example: Start a Workflow Using a UNIX Job ...................................................................... 53
  Example: Stop a Running Workflow Using a Windows Job .................................................. 53
Integration with CA Workload Automation CA 7 Edition ..................................................... 54
  Example: List the Workflows that Match a Specified Pattern Using a Windows Job .................. 54
  Example: Start a Workflow Using a UNIX Job ...................................................................... 55
  Example: Stop a Running Workflow Using a Windows Job .................................................. 55
Chapter 1: Overview

This section provides an overview of the Command Line Interface (CLI).

This section contains the following topics:

- **About This Guide** (see page 7)
- **Using the Command Line Interface** (see page 7)
- **Memory Limitations** (see page 9)
- **Workflow Run ID Not Supported** (see page 9)

About This Guide

This guide describes how to use the Command Line Interface (CLI) to integrate with Informatica PowerCenter. The guide is intended for scheduling developers and testers.

The CLI lets you invoke an agent workload that your scheduling manager does not currently support. For example, if your scheduling manager does not support Informatica jobs, you can use the CLI to run this workload as Windows or UNIX (command) jobs. In a future release of your scheduling manager, an Informatica job type will be added to support this workload. Then, you can upgrade your scheduling manager to take advantage of the new job type or continue to use the CLI.

**Note:** For information about running an agent workload using the supported job types, see the documentation for your scheduling manager.

Using the Command Line Interface

The CLI is a standalone utility that is shipped with the agent as a shell script (Linux) or batch script (Windows). To use the CLI, you define a UNIX or Windows (command) job in your scheduling manager. In the job definition, you specify the name of the shell or batch script and pass the necessary arguments.

The CLI writes all of the output data to standard output (stdout). When run as a UNIX or Windows (command) job, the agent captures the output into the job's spool file.
For example, to start an Informatica workflow, you pass the command name, repository folder name, workflow name, and other arguments. When the job is submitted, the shell or batch script runs on the agent and starts the workflow residing on the Informatica server. The response is stored in the job’s spool file.

The following diagram shows how to use the CLI to run an Informatica server job:
Memory Limitations

When using the CLI, the number of jobs that you can submit and monitor simultaneously is limited to the available free memory. For every CLI job, the agent launches a separate JVM, which results in extra overhead when compared to a dedicated job type supported on a scheduling manager. Generally, each CLI job consumes and occupies about 40 MB of memory until it finishes execution. For example, if a CLI job runs for 2 hours and is tracked to completion, the job occupies 40 MB of memory for the full 2 hours.

When the number of concurrent jobs that is submitted through the CLI consumes all the physical memory on the agent computer, the operating system is forced to swap memory to the hard drive. As a result, the latency between the time the job is submitted to the agent and the time the job starts to run increases. For example, the latency can increase from 1 second to more than 1 minute.

Notes:

■ To limit the number of jobs that are allowed to run at the same time, you can define initiators per job class on the agent. For more information about defining job classes and initiators on the agent, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

■ To use initiators, you also must specify the job class in the job definition on your scheduling manager. For more information about specifying the job class in the job definition, see your scheduling manager documentation.

■ Limiting initiators affects other workload running on the agent. When running CLI jobs, we recommend that you use a dedicated agent.

Workflow Run ID Not Supported

Due to a known issue in Informatica, the workflowrunID argument is not supported. As a result, you cannot identify the workflow run in the CLI command. When you retrieve status or log information about a workflow, the agent can only retrieve information about the last run of the workflow or its instance. Similarly, the agent can only monitor or stop the last run of the workflow or its instance.

Note: Due to this issue, you cannot run workflow instances in parallel. The run ID that is displayed in Informatica for a specified workflow may not match the run ID that is displayed in the spool file.
Chapter 2: Integrating with Informatica PowerCenter

CA WA Agent for Informatica supports integration with Informatica PowerCenter. In Informatica, jobs are known as workflows. Using the agent, you can start, monitor, and control Informatica workflows.

Informatica PowerCenter provides an environment that lets you load data into a centralized location, such as a data warehouse or operational data store (ODS). You can extract data from multiple sources, transform the data according to business logic you build in the client application, and load the transformed data into file and relational targets.

This section contains the following topics:

- Informatica Command Line Interface (see page 11)
- Invoking the Informatica CLI (see page 12)
- informatica Script—Integrate with Informatica PowerCenter (see page 13)
- Exit Codes for Job Submission (see page 39)

Informatica Command Line Interface

The Informatica CLI is a standalone utility that is shipped with the agent. You can use the CLI to run Informatica jobs that your scheduling manager does not currently support.

On Linux, the Informatica CLI is a shell script. You can run the shell script from the command line or you can schedule it to run using your scheduling manager, similar to other shell scripts. For example, you can define a UNIX (command) job to start an Informatica workflow using the CLI.

On Windows, the Informatica CLI is a batch script. You can run the batch script from the command line or you can schedule it to run using your scheduling manager, similar to other batch scripts. For example, you can define a Windows (command) job to start an Informatica workflow using the CLI.
Invoking the Informatica CLI

You can invoke the Informatica CLI to integrate the agent with Informatica PowerCenter.

The Informatica CLI (informatica) is located in the following directory:

- On Linux:
  
  \[install\_dir/\texttt{wrappers/informa\texttt{t}}\texttt{tica}\\

- On Windows:
  
  \[install\_dir/\texttt{wrappers/informa\texttt{tica.bat}\\

\texttt{install\_dir}  

  Specifies the agent installation directory.

To invoke the CLI, specify the following command:

\texttt{wrappers/informa\texttt{tica --argument ...\\

\texttt{--argument}  

  Specifies a CLI argument. Prefix each argument with two dashes (\texttt{--}).

Notes:

- By default, \texttt{install\_dir} is the working directory. Do not change the working directory when invoking the Informatica CLI.

- Enclose any values in the CLI that contain special characters in double quotes. For example, enclose any values that contain \% on Windows and \& on Linux in double quotes.
informatica Script—Integrate with Informatica PowerCenter

The informatica script provides integration with Informatica PowerCenter.

The script has the following format:

```bash
informatica [--verbose=true|false] [--infaTarget=infa_target_name] 
[--repositoryName=repository_name] 
[--user=user_name] [--password=password] --command=command 
--folderName=repository_folder_name --keyword=value ...
```

**--verbose=true|false**

(Optional) Indicates whether the agent logs the detailed messages for the step-by-step execution of the command in the spool files. The additional information logged is received from the agent and is not sent by Informatica.

**true**

Logs the detailed messages for the step-by-step execution of the command in the spool files.

**false**

Displays only the final output in the job spool file.

**Default**: false

**--infaTarget=infa_target_name**

(Optional) Specifies the name of the Informatica target (`infaTarget`). The `install_dir/config/informatica/infaTarget` directory contains the properties for connecting to the Informatica server and one or more database repositories.

**Note**: If this argument is not specified, the agent uses the default value specified in the `informatica.server.target.default` parameter in the `agentparm.txt` file.

**--repositoryName=repository_name**

(Optional) Specifies the name of the repository created on the Informatica server. The repository can be obtained from the Informatica Administrative console. The `repositoryName.properties` file for a specified Informatica target (`infaTarget`) contains the database connection properties.

**Limits**: Up to 128 characters; cannot contain delimiters (such as spaces) and special characters

**Notes**: If this argument is not specified, the agent uses the default value specified in the `informatica.repository.name.default` parameter in the `informatica.properties` file.
--user=user_name

(Optional) Specifies the Informatica server user name.

Notes:

■ If this argument is not specified, the agent uses the default value specified in the informatica.user.id.default parameter in the informatica.properties file.

■ If you specify the user argument, you must also specify the password argument.

--password=password

(Optional) Specifies the password for the Informatica server user name.

Important! To avoid specifying passwords in job definitions, omit the user and password arguments and use the default user settings specified in the informatica.properties file. We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

Notes:

■ The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

■ If this argument is not specified, the agent uses the default value specified in the informatica.password.default parameter in the informatica.properties file

--command=command

Specifies the command that you want to run. Options are as follows:

startWorkflow

Starts the specified workflow or its instance from the beginning of the workflow or from a specified task.

listAllWorkflows

Lists workflows defined in Informatica that match specified filtering criteria. The command returns a table that contains the following columns:

■ Workflow name

■ Workflow IDs

■ Workflow validity

■ Workflow instance names
listWorkflowTasks

Lists all the tasks within a specified workflow. The command returns a table that contains the following columns:

- Workflow name
- Task name
- Task type

getWorkflowDetails

Displays the sequence of tasks for a specified Informatica workflow and the details of the links connecting the tasks.

getObjectHierarchyInWorkflow

Retrieves the hierarchy of objects within a specified workflow. The command returns a table that contains the following columns:

- Workflow name
- Workflow ID
- Workflow hierarchy

getTaskDetails

Displays the details of a task within a specified workflow. The command returns the following details about the task:

- Task type
- Mapping name
- Folder ID
- Workflow ID
- Workflow name
- Task instance ID
- Task instance name

getWorkflowStatus

Retrieves the status of a specified workflow or its instance.

**Note:** The status of the last run of the workflow or its instance is retrieved.

getWorkflowlog

Retrieves the log information of the last run of a specified workflow or its instance or its sessions.

**Note:** The log of the last run of the workflow or its instance is retrieved.
getWorkflowRunDetails
Displays the details of the last run of a specified workflow or its instance. The following details are included:

- Workflow start time
- Workflow end time
- Workflow total run time
- Workflow status (including error code and messages in case of workflow failure)
- Start time of tasks in the workflow
- End time of tasks in the workflow
- Total run time of tasks in the workflow
- Status of tasks (including error code and messages in case of task failure)
- Task type

**Note:** The details of the last run of the workflow or its instance is retrieved.

monitorWorkflow
Monitors continuously a running workflow and its corresponding tasks. If the workflow is not running, the command provides the last run details of the workflow and all its tasks.

**Note:** The last run of the workflow or its instance is monitored.

restartWorkflow
Restarts a failed workflow or its instance from the beginning (similar to startWorkflow). Alternatively, if you specify a session task, the command recovers a workflow or its instance from a point of failure.

stopWorkflow
Stops a running workflow or its instance.

**Note:** The last run of the workflow or its instance is stopped.

`-- folderName=repository_folder_name`

Specifies the Informatica PowerCenter repository folder name where the specified workflow resides. You can obtain this value from the Informatica server.

`-- keyword=value`

Specifies keywords for the command.
The startWorkflow command contains the following keywords:

```
--workflowName=workflow_name
```

Specifies the name of the PowerCenter workflow to be started. You can get the list of available workflows using the listAllWorkflows command.

```
--workflowInstanceName=workflow_instance_name
```

(Optional) Specifies the name of the workflow instance to be started. When the task name and the workflow instance name are specified, the workflow instance is started from the specified task. When the task name is not specified, the workflow instance starts from the beginning. If the instance name is not specified, the workflow is started. You can get a list of workflow instances using the listAllWorkflows command.

```
--taskName=task_name
```

(Optional) Specifies the name of the task to start the workflow from. If the task name is not specified, the workflow starts from the beginning. You can get a list of available tasks within a workflow using the listWorkflowTasks command.

```
--track=true|false
```

(Optional) Indicates whether to track the job to completion.

**true**

Tracks the job to completion. The CLI call returns when the job completes, whether successfully or not. The task and task step execution status is written to the spool file. The completion code of the CLI call indicates whether the job failed or succeeded. If the job was submitted successfully and the job completed successfully, 0 is returned. If job submission failed or the job failed during monitoring, a non-zero code is returned.

**Note:** Since the agent launches one JVM for every CLI call, the JVM overhead can be substantial when too many CLI calls are running concurrently. If the job takes a considerable amount of time to finish, system resources can be impacted.

**false**

Does not track the job to completion. The agent exits the JVM immediately after the job is submitted instead of waiting for the job completion. To retrieve the job running status, you can use the GetWorkflowStatus operation. The completion code of the CLI call indicates only whether the job submission succeeded or failed. If the job was submitted successfully, 0 is returned. If job submission failed, a non-zero code is returned. The spool file indicates the reason for the job submission error, such as a JDBC connection failure.

**Default:** true
--passOnSuccessOnly=true | false

(Optional) Indicates whether the status of the workflow is retrieved directly from Informatica or inferred depending on the successful execution of all the tasks. This argument is only used when the track argument is set to true.

true

Returns the workflow status based on the status of tasks in the Informatica workflow. If all tasks in the workflow complete successfully, the agent reports the status of the workflow as successful. If any task in the workflow fails, the agent reports the status of the workflow as failed.

false

Returns the workflow status as reported by Informatica. In Informatica, a workflow can succeed even if one or more tasks in the workflow fail.

Default: false

Notes:

- The default value of the flag is read from the informatica.passonsuccessonly.default parameter in the agentparm.txt file.
- If the passOnSuccessOnly flag is not specified in the job definition or configuration file, the default value is false.

--paramFile=parameter_file

(Optional) Specifies the path and name of a parameter file on the Informatica server. Informatica PowerCenter processes the parameter file when running a workflow. The parameter file contains values for symbolic variables and must adhere to the format that Informatica PowerCenter expects. A property value that is set in a parameter file overrides a value for this property that is set at the session level or elsewhere.

The listAllWorkflows command contains the following keywords:

--matchingPattern="workflow_pattern"

(Optional) Specifies the matching pattern string that is used to retrieve the names of the workflows. If you specify a matching pattern, only workflows whose names contain the search pattern are displayed. If you do not specify a matching pattern, all workflows that belong to the specified repository folder are displayed. % is the only wildcard character that is supported for the pattern matching.

Example: If you specify "A%ab%" as the matching pattern, all the workflows whose names start with A and contain ab in the middle or at the end are displayed. If you specify "%file%" as the matching pattern, all the workflows whose names contain file are displayed, for example: file_task, X_file_Task and T_file.
The listWorkflowTasks command contains the following keywords:

```
--workflowName=workflow_name
```

Specifies the name of the workflow that contains the tasks you want to list. You can list the available workflows using the listAllWorkflows command.

The getWorkflowDetails command contains the following keywords:

```
--workflowName=workflow_name
```

Specifies the name of the workflow that you want to display the details for. You can get the list of available workflows using the listAllWorkflows command.

The getObjectHierarchyInWorkflow command contains the following keywords:

```
--workflowName=workflow_name
--taskName=task_name
```

Specifies the name of the workflow that you want to display the object hierarchy for. You can get the list of available workflows using the listAllWorkflows command. (Optional) Specifies the name of the task that you want to display the object hierarchy for. You can get the list of available tasks within a workflow using the listWorkflowTasks command.

The getTaskDetails command contains the following keywords:

```
--workflowName=workflow_name
--taskName=task_name
```

Specifies the name of the workflow that you want to display the task details for. You can get the list of available workflows using the listAllWorkflows command. Specifies the name of the task that you want to display the details for. You can get the list of available tasks within a workflow using the listWorkflowTasks command.

The getWorkflowStatus command contains the following keywords:

```
--workflowName=workflow_name
--workflowInstanceName=workflow_instance_name
```

Specifies the name of the workflow that you want to retrieve the status for. You can get the list of available workflows using the listAllWorkflows command. (Optional) Specifies the name of the workflow instance that you want to retrieve the status for. If the instance name is specified, the status of that instance in its last run is displayed. If the instance name is not specified, the workflow status of its last run is displayed. You can get the list of workflow instances using the listAllWorkflows command.
**passOnSuccessOnly=true** | false

(Optional) Indicates whether the status of the workflow is retrieved directly from Informatica or inferred depending on the successful execution of all the tasks. This argument is only used when the track argument is set to true.

**true**

Returns the workflow status based on the status of tasks in the Informatica workflow. If all tasks in the workflow complete successfully, the agent reports the status of the workflow as successful. If any task in the workflow fails, the agent reports the status of the workflow as failed.

**false**

Returns the workflow status as reported by Informatica. In Informatica, a workflow can succeed even if one or more tasks in the workflow fail.

**Default:** false

**Notes:**

- The default value of the flag is read from the informatica.passonsuccessonly.default parameter in the agentparm.txt file.
- If the passOnSuccessOnly flag is not specified in the job definition or configuration file, the default value is false.

The `getWorkflowLog` command contains the following keywords:

**--workflowName=** workflow name

Specifies the name of the workflow that you want to display the log for. You can get the list of available workflows using the `listAllWorkflows` command.

**--workflowInstanceName=** workflow instance name

(Optional) Specifies the name of the workflow instance that you want to display the log for. If the instance name is specified, the logs of that instance in its last run are displayed. If the instance name is not specified, the workflow log of its last run is displayed. You can get the list of workflow instances using the `listAllWorkflows` command.

**--sessionName=** session name

(Optional) Specifies the name of the session within the workflow that you want to display the log for. If the session name is not specified, only the workflow logs are displayed. You can get the list of available sessions using the `listWorkflowTasks` command.
The `getWorkflowRunDetails` command contains the following keywords:

--- `--workflowName=workflow_name`

Specifies the name of the workflow that you want to display the run details for. You can get the list of available workflows using the `listAllWorkflows` command.

--- `--workflowInstanceName=workflow_instance_name`

(Optional) Specifies the name of the workflow instance that you want to display the run details for. If the instance name is not specified, the details of its last run of the workflow is displayed. You can get the list of workflow instances using the `listAllWorkflows` command.

The `monitorWorkflow` command contains the following keywords:

--- `--workflowName=workflow_name`

Specifies the name of the workflow that you want to monitor. You can get the list of available workflows using the `listAllWorkflows` command.

--- `--workflowInstanceName=workflow_instance_name`

(Optional) Specifies the name of the workflow instance that you want to monitor. If the instance name is not specified, the agent monitors the workflow. You can get a list of workflow instances using the `listAllWorkflows` command.

--- `--passOnSuccessOnly=true|false`

(Optional) Indicates whether the status of the workflow is retrieved directly from Informatica or inferred depending on the successful execution of all the tasks. This argument is only used when the `track` argument is set to true.

- **true**

  Returns the workflow status based on the status of tasks in the Informatica workflow. If all tasks in the workflow complete successfully, the agent reports the status of the workflow as successful. If any task in the workflow fails, the agent reports the status of the workflow as failed.

- **false**

  Returns the workflow status as reported by Informatica. In Informatica, a workflow can succeed even if one or more tasks in the workflow fail.

**Default:** false

**Notes:**

- The default value of the flag is read from the `informatica.passonsuccessonly.default` parameter in the `agentparm.txt` file.

- If the `passOnSuccessOnly` flag is not specified in the job definition or configuration file, the default value is false.
The restartWorkflow command contains the following keywords:

--workflowName=workflow_name

Specifies the name of the workflow that you want to restart. If the task name is not specified, the workflow is restarted from the beginning in recovery mode. You can get the list of available workflows using the listAllWorkflows command.

--workflowInstanceName=workflow_instance_name

(Optional) Specifies the name of the workflow instance that you want to restart. If the task name is not specified, the specified instance of the workflow is restarted from the beginning in recovery mode. You can get a list of workflow instances using the listAllWorkflows command.

--taskName=task_name

(Optional) Specifies the name of the session task that you want to restart the workflow from. The workflow or workflow instance is restarted in recovery mode from the specified session task. You can get the list of available tasks within a workflow using the listWorkflowTasks command.

Note: To restart a workflow or its instance from a point a failure, specify a session task. If a non-session task is specified, the agent reports an error.

--track=true|false

(Optional) Indicates whether to track the job to completion.

true

Tracks the job to completion. The CLI call returns when the job completes, whether successfully or not. The task and task step execution status is written to the spool file. The completion code of the CLI call indicates whether the job failed or succeeded. If the job was submitted successfully and the job completed successfully, 0 is returned. If job submission failed or the job failed during monitoring, a non-zero code is returned.

Note: Since the agent launches one JVM for every CLI call, the JVM overhead can be substantial when too many CLI calls are running concurrently. If the job takes a considerable amount of time to finish, system resources can be impacted.

false

Does not track the job to completion. The agent exits the JVM immediately after the job is submitted instead of waiting for the job completion. To retrieve the job running status, you can use the GetWorkflowStatus operation. The completion code of the CLI call indicates only whether the job submission succeeded or failed. If the job was submitted successfully, 0 is returned. If job submission failed, a non-zero code is returned. The spool file indicates the reason for the job submission error, such as a JDBC connection failure.

Default: true
--passOnSuccessOnly=true | false

(Optional) Indicates whether the status of the workflow is retrieved directly from Informatica or inferred depending on the successful execution of all the tasks. This argument is only used when the track argument is set to true.

true

Returns the workflow status based on the status of tasks in the Informatica workflow. If all tasks in the workflow complete successfully, the agent reports the status of the workflow as successful. If any task in the workflow fails, the agent reports the status of the workflow as failed.

false

Returns the workflow status as reported by Informatica. In Informatica, a workflow can succeed even if one or more tasks in the workflow fail.

Default: false

Notes:

- The default value of the flag is read from the informatica.passonsuccessonly.default parameter in the agentparm.txt file.
- If the passOnSuccessOnly flag is not specified in the job definition or configuration file, the default value is false.

--paramFile=parameter_file

(Optional) Specifies the path and name of a parameter file on the Informatica server. Informatica PowerCenter processes the parameter file when running a workflow. The parameter file contains values for symbolic variables and must adhere to the format that Informatica PowerCenter expects. A property value that is set in a parameter file overrides a value for this property that is set at the session level or elsewhere.

The stopWorkflow command contains the following keywords:

--workflowName=workflow_name

Specifies the name of the running workflow that you want to stop. You can get the list of available workflows using the listAllWorkflows command.

--workflowInstanceName=workflow_instance_name

(Optional) Specifies the name of the workflow instance that you want to stop. You can get the list of workflow instances using the listAllWorkflows command.
Example: Start a Workflow on Informatica from a Task

This example starts a workflow named wf_inst on the Informatica server. The workflow is started from the cmd_copy_file_inst_a task.

wrappers/informatica.bat --infaTarget=tick --user=u_japi --password=32704AC768A31370 --command=startWorkflow --folderName=ipn --workflowName=wf_inst --taskName=cmd_copy_file_inst_a

The output is similar to the following:

----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 15:41:56 2012
----------------------------------------------------------------
Workflow Id: 34
Workflow Run Id: 33786
Status of workflow wf_inst after sending the command startWorkflow is running

Example: Start a Workflow on Informatica from the Beginning

This example starts a workflow named wf_inst on the Informatica server from the beginning:

wrappers/informatica.bat --infaTarget=tick --user=u_japi --password=32704AC768A31370 --command=startWorkflow --folderName=ipn --workflowName=wf_inst

The output is similar to the following:

----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 15:41:56 2012
----------------------------------------------------------------
Workflow Id: 34
Workflow Run Id: 33786
Status of workflow wf_inst after sending the command startWorkflow is running
Example: Start a Workflow on Informatica with a Parameter File

This example starts a workflow named WF_3 on the Informatica server. When the workflow runs, Informatica PowerCenter processes the specified parameter file.

```
wrappers/informatica.bat --command=startWorkflow --folderName=INFAPLUGIN
--workflowName=WF_3 --paramFile=C:/Infa/wf3.prm
```

The wf3.prm parameter file includes symbolic variables that override the default log directory and file name:

```
;Workflow Parameters
[INFAPLUGIN.WF:wf_3]
;Session parameters
$PMSessionLogFile=wf3.log
$PMSessionLogDir=C:\Infa\Log
;mapping parameters
```

For this workflow, log files will be stored in the C:\Infa\Log directory on the Informatica server and their name will start with wf3.log. The actual files that Informatica generates will be longer, for example, wf3.log.1358.20130726150952.bin.

Example: List Workflows that Match a Specified Pattern

This example lists all workflows in the ipn folder that contain wf in their name:

```
wrappers/informatica.bat --repositoryName=infa_rep_srv
--user=u_japi --password=32704AC768A31370 --command=listAllWorkflows
--folderName=ipn --matchingPattern=%wf%_%
```

The output is similar to the following:

```
---------------------------------------------------------
Workflow WorkflowId Status Instance Names
---------------------------------------------------------
WF_3 23 Valid RUNINSTANCE1
WF_EMP 7 Valid RUNINSTANCE1, RUNINSTANCE2
WF_LOD 2 Valid
WF2_INST17 Valid
SWF_2 20 Valid
WF3_NRUN27 Valid
---------------------------------------------------------
```
Example: List All Workflows in a Folder

This example lists all workflows in the ipn folder:

```
wrappers/informatica.bat --command=listAllWorkflows --folderName=ipn
```

The output is similar to the following:

```
-----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 15:53:37 2012
-----------------------------------------------------------------
Workflow List
****List of Workflows in folder IPN****
Workflow Workflow Id Status Instances
-----------------------------------------------------------------
WF_PERF  68   Valid   RUNINSTANCE1
WF_NEW1  78   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_NEW   27   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_INST  34   Valid   RUNINSTANCE1, RUNINSTANCE2, RUNINSTANCE3, RUNINSTANCE4
WF_SP    38   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_MY_TASKS  54   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_16OCT 109  Valid   RUNINSTANCE1, RUNINSTANCE2
WF_S1_NEW 115  Valid
WF_STOP  82   Valid   RUNINSTANCE1, RUNINSTANCE2, RUNINSTANCE3, RUNINSTANCE4
WF_FILE_COPY_2  98   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_NEVER_RUN 60   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_1    3    Valid   RUNINSTANCE1
WF_2    6    Valid   RUNINSTANCE1, RUNINSTANCE2
WF_3    21   Valid
WF_STOP1 84   Valid   RUNINSTANCE1, RUNINSTANCE2
MY_NEW_WORKFLOW 73   Valid
WF_NRUN  52   Invalid
WF_HTTP  66   Valid   RUNINSTANCE1, RUNINSTANCE2
WF_EVENT 104  Valid   RUNINSTANCE1, RUNINSTANCE2
WF_FILE_COPY_3 101  Valid
WF_FILE_COPY_1  95  Valid   RUNINSTANCE1, RUNINSTANCE2
```
Example: Display All Tasks for a Workflow

This example displays all tasks for a workflow named wf_2 in the ipn folder:

```
wrappers/informatica.bat --command=listWorkflowTasks
    --folderName=ipn --workflowName=wf_2
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 16:10:40 2012
----------------------------------------------------------------
Workflow Tasks
List of tasks present in the workflow wf_2
Workflow Id: 6
Workflow Name Task(Task Type)
wf_2 CMD_COPY_FILE_B1(COMMAND)
wf_2 CMD_COPY_FILE_B(COMMAND)
wf_2 CMD_COPY_FILE_A(COMMAND)
wf_2 TIM_A(TIMER)
```

Example: Display the Sequence of Tasks in a Workflow

This example displays the sequence of tasks in a workflow named wf_1 in the ipn folder:

```
wrappers/informatica.bat --command=getWorkflowDetails
    --folderName=ipn --workflowName=wf_1
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 16:12:18 2012
----------------------------------------------------------------
Attribute Details
Workflow Id: 3
Workflow Name: wf_1
Start --> s_m_copy_txt_file
s_m_copy_txt_file --> wl_level_A <Link> $s_m_copy_txt_file.PrevTaskStatus=Succeeded
wl_level_A --> cmd_copy_new_file <Link> $wl_level_A.PrevTaskStatus=Succeeded
cmd_copy_new_file --> TIM_A
```
**Example: Display the Sequence of the Tasks in a Workflow Using Verbose**

This example displays the sequence of tasks in a workflow named `wf_1` in the `ipn` folder when the verbose option is set to true:

```
wrappers/informatica.bat --verbose=true --command=getWorkflowDetails --folderName=ipn --workflowName=wf_1
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 16:13:10 2012
----------------------------------------------------------------
Validating the input parameters
Validating length of input parameters
Creating connection with informatica server
Creating database connection with informatica repository
Execution starts to fetch the attribute details of workflow
Validating the folder
Validating the workflow
Displaying from task and to task starting from START Task
Attribute Details
Workflow Id: 3
Workflow Name: wf_1
Start --> s_m_copy_txt_file
s_m_copy_txt_file --> wl_level_A <Link> $s_m_copy_txt_file.PrevTaskStatus=Succeeded
wl_level_A --> cmd_copy_new_file <Link> $wl_level_A.PrevTaskStatus=Succeeded
cmd_copy_new_file --> TIM_A
Task Details along with their sequences displayed successfully
```
**Example: Display the Hierarchy of a Task Within a Workflow**

This example displays the hierarchy of a task within a workflow:

```
wrappers/informatica.bat --command=getObjectHierarchyInWorkflow
--folderName=ipn --workflowName=wf_1 --taskName=s_m_Copy_Txt_File
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 16:18:03 2012
----------------------------------------------------------------
Object Hierarchy
Workflow Name: wf_1
Workflow Id: 3
Hierarchy of objects is
wf_1-->S_M_COPY_TXT_FILE
```
Example: Display the Hierarchy of a Workflow Using Verbose

This example displays the hierarchy of all the objects in the workflow when the verbose option is set to true:

wrappers/informatica.bat --command=getObjectHierarchyInWorkflow --folderName=ipn --workflowName=wf_1 --verbose=true

The output is similar to the following:

------------------------------------------------------------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 16:15:40 2012
------------------------------------------------------------------------------------------------------------------
Validating the input parameters
Validating length of input parameters
Creating connection with informatica server
Creating database connection with informatica repository
Execution starts to fetch the hierarchy of objects in workflow wf_1
Validating the folder
Validating the workflow
Fetching the hierarchy of all objects in workflow wf_1
Object Hierarchy
Workflow Name: wf_1
Workflow Id: 3
Hierarchy of objects is
wf_1-->START
wf_1-->TIM_A
wf_1-->CMD_COPY_FILE
wf_1-->S_M_COPY_TXT_FILE
wf_1-->WL_LEVEL_A
wf_1-->WL_LEVEL_A--->START
wf_1-->WL_LEVEL_A--->CMD_COPY_FILE_INST_A
wf_1-->WL_LEVEL_A--->CMD_COPY_FILE_INST_B
wf_1-->WL_LEVEL_A--->CMD_COPY_WL_1
wf_1-->WL_LEVEL_A--->WL_LEVEL_B
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->START
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->CMD_COPY_WL_2
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->WL_LEVEL_C
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->WL_LEVEL_C--->START
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->WL_LEVEL_C--->CMD_COPY_WL_3
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->WL_LEVEL_C--->WL_LEVEL_D
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->WL_LEVEL_C--->WL_LEVEL_D--->START
wf_1-->WL_LEVEL_A--->WL_LEVEL_B--->WL_LEVEL_C--->WL_LEVEL_D--->CMD_COPY_WL_4
Hierarchy for the given objects retrieved successfully
Example: Display the Details of a Task Within a Workflow

This example displays the details of a task named s_m_copy_txt_file within the wf_1 workflow:

```
wrappers/informatica.bat --command=getTaskDetails --folderName=ipn
--workflowName=wf_1 --taskName=s_m_copy_txt_file --infaTarget=tick
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 18:19:43 2012
----------------------------------------------------------------
Task Details
Task Type: session_task
Mapping Name: m_copy_txt_file
Folder Id: 2
Folder Name: IPN
Workflow Id: 34
Workflow Name: wf_inst
Task Instance Id: 52
Task Instance Name: s_m_copy_txt_file
```

Example: Retrieve the Status of a Workflow Instance Using Verbose

This example retrieves the status of a workflow instance named runinstance1 of the wf_inst workflow when the verbose option is set to true:

```
wrappers/informatica.bat --verbose=true --command=getWorkflowStatus
--folderName=ipn
--workflowName=wf_inst --passOnSuccessOnly=true
--workflowInstanceName=runinstance1
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 18:06:11 2012
----------------------------------------------------------------
Workflow Status
Workflow Id: 34
Workflow Run Id: 33797
Workflow Name: wf_inst
Workflow Status: succeeded
```
Example: Retrieve the Log of a Workflow Instance Using Verbose

This example retrieves the log of a workflow instance named runinstance1 of the wf_inst workflow when the verbose option is set to true:

```
wrappers/informatica.bat --verbose=true --command=getWorkflowLog --folderName=ipn --workflowName=wf_inst --workflowInstanceName=runinstance1
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 18:03:37 2012
----------------------------------------------------------------
Log Type :WorkflowLogs
Workflow Log
Workflow Id: 34
Workflow Run Id: 33797
Log Type :WorkflowLogs
INFO : Fri Dec 14 07:12:21 2012 [LM_44206] : 2496 Workflow wf_inst started with run id [33797], run instance name [], run type [Concurrent Run with Same Instance Name].


INFO : Fri Dec 14 07:12:27 2012 [LM_36488] : 5240 Session task instance [s_m_copy_txt_file] : [TM_6792 Notifying the Integration Service that the prepare phase has been completed. : (Fri Dec 14 07:12:27 2012)]
INFO : Fri Dec 14 07:12:30 2012 [LM_36505] : 2496 Link [s_m_copy_txt_file --> TIM_1]: empty expression string, evaluated to TRUE.
INFO : Fri Dec 14 07:12:30 2012 [LM_36609] : 2496 Timer task instance [TIM_1]: Timer task specified to wait [0 days, 0 hours, 8 minutes, 0 seconds] from start of this task.
```
INFO : Fri Dec 14 07:12:30 2012 [LM_36606] : 2496 Timer task instance [TIM_1]: The timer will complete at [Fri Dec 14 07:20:30 2012].
INFO : Fri Dec 14 07:20:30 2012 [LM_36605] : 280 Timer task instance [TIM_1]: The timer task has finished.

Workflow/Session log fetched Successfully
Example: Display the Run Details of a Workflow

This example retrieves the run details of the wf_inst workflow:

```
wrappers/informatica.bat --command=getWorkflowRunDetails --folderName=ipn
--workflowName=wf_inst
```

The output is similar to the following:

```
----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 17:59:31 2012
----------------------------------------------------------------
Workflow Run Details
Workflow Name: wf_inst
Workflow Id: 34
Workflow Run Id: 33797
Workflow Start Time: 14-12-2012 7:12:21
Workflow End Time: 14-12-2012 7:20:30
Workflow Status: succeeded
Workflow Run Time 08 minutes 09 seconds
Run Error Code 0
Run Error Message: NA
Task Name: cmd_copy_File_INST_A
Task Type: command_task
Task Start Time: 14-12-2012 7:12:21
Task End Time: 14-12-2012 7:12:22
Task Run Time 01 seconds
Task Run Error Code 0
Task Run Error Message: NA
Task Status: succeeded
Task Name: cmd_copy_File_INST_B
Task Type: command_task
Task Start Time: 14-12-2012 7:12:22
Task End Time: 14-12-2012 7:12:24
Task Run Time 02 seconds
Task Run Error Code 0
Task Run Error Message: NA
Task Status: succeeded
Task Name: cmd_copy_File_INST_C
Task Type: command_task
Task Start Time: 14-12-2012 7:12:24
Task End Time: 14-12-2012 7:12:26
Task Run Time 02 seconds
Task Run Error Code 0
Task Run Error Message: NA
Task Status: succeeded
Task Name: s_m_copy_txt_file
Task Type: session_task
Task Start Time: 14-12-2012 7:12:26
```
Informatica Script—Integrate with Informatica PowerCenter

Task End Time: 14-12-2012 7:12:28
Task Run Time 02 seconds
Task Run Error Code 0
Task Run Error Message: NA
Task Status: succeeded
Task Name: TIM_1
Task Type: timer_task
Task Start Time: 14-12-2012 7:12:30
Task End Time: 14-12-2012 7:20:30
Task Run Time 08 minutes
Task Run Error Code 0
Task Run Error Message: NA
Task Status: succeeded
Example: Monitor a Workflow Using Verbose

This example monitors a running wf_inst workflow when the verbose option is set to true:

```
wrappers/informatica.bat --verbose=true --command=monitorWorkflow
--folderName=ipn --workflowName=wf_inst
```

The output is similar to the following:

```
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 17:42:07 2012

Workflow Id :34
Workflow Run Id :33797
Status Updated At 2012-12-14 17:42:36.112
Workflow Id :34
Workflow Run Id :33797
Current status of the Workflow wf_inst is : running
Current status of task TIM_1 is : running
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Status Updated At 2012-12-14 17:43:17.612
Workflow Id :34
Workflow Run Id :33797
Current status of the Workflow wf_inst is : running
Current status of task TIM_1 is : running
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Status Updated At 2012-12-14 17:44:03.722
Workflow Id :34
Workflow Run Id :33797
Current status of the Workflow wf_inst is : running
Current status of task TIM_1 is : running
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Status Updated At 2012-12-14 17:45:26.628
Workflow Id :34
Workflow Run Id :33797
Current status of the Workflow wf_inst is : running
Current status of task TIM_1 is : running
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Status Updated At 2012-12-14 17:46:09.628
```

---
Current status of task cmd_copy_File_INST_A is : succeeded  
Current status of task cmd_copy_File_INST_C is : succeeded  
Status Updated At 2012-12-14 17:50:11.456  
Workflow Id :34  
Workflow Run Id :33797  
Current status of the workflow wf_inst is : succeeded  
Current status of task cmd_copy_File_INST_B is : succeeded  
Current status of task s_m_copy_txt_file is : succeeded  
Current status of task TIM_1 is : succeeded  
Current status of task cmd_copy_File_INST_A is : succeeded  
Current status of task cmd_copy_File_INST_C is : succeeded  
--------Workflow Status --------  
Workflow Name :wf_inst  
Status :succeeded  
Workflow Error Code :0  
Workflow Error Message :NA  
Monitoring of Workflow completed successfully

Example: Restart a Workflow Instance Without Tracking

This example restarts a workflow instance named runinstance1 of the wf_inst workflow when the track option is set to false by default:

wrappers/informatica.bat --command=restartWorkflow --folderName=ipn  
--workflowName=wf_inst --workflowInstanceName=runinstance1

The output is similar to the following:

----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN  
Start date Fri Dec 14 17:24:18 2012  
----------------------------------------------------------------
Workflow Id :34  
Workflow Run Id :33795  
Status after sending restartworkflow command of the instance of runinstance1 of workflow wf_inst is running
Example: Restart a Workflow Using Tracking

This example restarts the wf_inst workflow when the track option is set to true:

wrappers/informatica.bat --command=restartWorkflow --folderName=ipn
--workflowName=wf_inst --track=true

The output is similar to the following:

----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 17:29:18 2012
----------------------------------------------------------------
Validating the input parameters
Validating length of input parameters
Workflow Id :34
Workflow Run Id :33796
Status after sending restartworkflow command of the workflow wf_inst is running
Status Updated At=2012-12-14 17:29:40.362
Current status of the workflow wf_inst is : running
Current status of task TIM_1 is : running
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Current status of task s_m_copy_txt_file is : succeeded
Status Updated At=2012-12-14 17:30:25.034
Current status of the workflow wf_inst is : running
Current status of task TIM_1 is : running
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Current status of task s_m_copy_txt_file is : succeeded
Status Updated At=2012-12-14 17:31:12.8
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Current status of task s_m_copy_txt_file is : succeeded
Status Updated At 2012-12-14 17:38:28.519
Current status of the workflow wf_inst is : succeeded
Current status of task cmd_copy_File_INST_B is : succeeded
Current status of task cmd_copy_File_INST_A is : succeeded
Current status of task cmd_copy_File_INST_C is : succeeded
Current status of task s_m_copy_txt_file is : succeeded
Workflow Status

Workflow Name :wf_inst
Status :succeeded
Workflow Error Code :0
Workflow Error Message :NA
Monitoring of Workflow completed successfully

Example: Stop a Running Workflow Instance

This example stops a running workflow instance named runinstance1 of the wf_inst workflow:

wrappers/informatica.bat --command=stopWorkflow --folderName=ipn
--workflowName=wf_inst --workflowInstanceName=runinstance1

The output is similar to the following:

----------------------------------------------------------------
Output of messages for workload object JOB/APPL.1/MAIN
Start date Fri Dec 14 16:25:58 2012
----------------------------------------------------------------
Stop Workflow Response
Workflow Id :34
Status of instance runinstance1 of workflow wf_inst after sending StopWorkflow command is stopped

Exit Codes for Job Submission

When submitting a job request or a command, the agent can return exit codes as described in the following table. An exit code of 0 implies success; all other exit codes are reported as errors.

Note: Not all exit codes apply to all CLI commands.

<table>
<thead>
<tr>
<th>Exit Code</th>
<th>Description</th>
<th>Submission/Execution Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Generic Exception</td>
<td>Execution Error</td>
</tr>
<tr>
<td>3</td>
<td>JLM Exception</td>
<td>Execution Error</td>
</tr>
<tr>
<td>4</td>
<td>Database-related exception</td>
<td>Execution Error</td>
</tr>
<tr>
<td>5</td>
<td>Communication exception</td>
<td>Execution Error</td>
</tr>
<tr>
<td>6</td>
<td>Cyb Persistence Exception</td>
<td>Submission Error</td>
</tr>
</tbody>
</table>
Exit Codes for Job Submission

<table>
<thead>
<tr>
<th>Exit Code</th>
<th>Description</th>
<th>Submission/Execution Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>CybInformatica Exception</td>
<td>Execution Error</td>
</tr>
<tr>
<td>8</td>
<td>Generic Failure</td>
<td>Execution Error</td>
</tr>
<tr>
<td>9</td>
<td>Interrupted Exception</td>
<td>Execution Error</td>
</tr>
<tr>
<td>10</td>
<td>Task Failed (PassOnSuccess check)</td>
<td>Execution Error</td>
</tr>
<tr>
<td>21</td>
<td>Incorrect argument</td>
<td>Submission Error</td>
</tr>
<tr>
<td>99</td>
<td>Properties file could not be loaded</td>
<td>Submission Error</td>
</tr>
<tr>
<td>99</td>
<td>Parameters not present in agentpram.txt</td>
<td>Submission Error</td>
</tr>
<tr>
<td>100</td>
<td>Parameters not present in the request</td>
<td>Submission Error</td>
</tr>
</tbody>
</table>

Exit Codes for Task Failures

When starting or restarting a workflow, the agent determines the exit code after completing the monitoring of the workflow as follows:

- If the workflow and all the included tasks complete successfully, the exit code is 0. No error messages are written to the spool file.
- If the workflow fails and only one task within the workflow fails, the exit code that is related to the failed task is returned. An error message that is related to the failed task is written to the spool file.
- If the workflow fails and more than one task within the workflow fails, a generic exit code and error message is returned, indicating multiple task failures within the workflow. The error code and error messages of all the failed tasks are written to the spool file.
- If the workflow completes successfully but one or more tasks within the workflow fails, the exit code is 0. The error code and error messages of the failed tasks are written to the spool file.

The error codes that Informatica returns for task failures have a wide range. To provide an appropriate exit code for every type of failure, the agent groups the types of task failures, as shown in the following table:

<table>
<thead>
<tr>
<th>Exit Code</th>
<th>Informatica Error Code Range</th>
<th>Informatica Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Generic error – does not belong to any category below</td>
</tr>
<tr>
<td>101</td>
<td></td>
<td>Multiple errors reported</td>
</tr>
<tr>
<td>Exit Code</td>
<td>Informatica Error Code Range</td>
<td>Informatica Error Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>102</td>
<td>ADV_13226</td>
<td>ADV Messages: Custom transformation library error</td>
</tr>
<tr>
<td>103</td>
<td>ALERT_10000 - ALERT_10010</td>
<td>ALERT Messages: Alert-related errors</td>
</tr>
<tr>
<td>104</td>
<td>ATHR_10000 - ATHR_10026</td>
<td>ATHR Messages: Service manager-related errors</td>
</tr>
<tr>
<td>105</td>
<td>AUTH_10000 - AUTH_10039</td>
<td>AUTH Messages: Authentication-related errors</td>
</tr>
<tr>
<td>106</td>
<td>AUTHEN_10001 &amp; AUTHEN_10002</td>
<td>AUTHEN Messages: Service manager authentication errors</td>
</tr>
<tr>
<td>107</td>
<td>BAPI_99101 - BAPI_99140</td>
<td>BAPI Messages: BAPI-related errors</td>
</tr>
<tr>
<td>108</td>
<td>BR_16001 - BR_16056</td>
<td>BR Messages: Database-related errors</td>
</tr>
<tr>
<td>109</td>
<td>BTree_90002 - BTree_90014</td>
<td>BTree Messages: Cache-related errors</td>
</tr>
<tr>
<td>110</td>
<td>BW_41013 - BW_41082</td>
<td>BW Messages: BW service-related errors</td>
</tr>
<tr>
<td>111</td>
<td>CFG_10000 - CFG_10116</td>
<td>CFG Messages: Configuration-related errors</td>
</tr>
<tr>
<td>112</td>
<td>CMD_35197 - CMD_35198</td>
<td>CMD Messages: Environment variable-related errors</td>
</tr>
<tr>
<td>113</td>
<td>CMN_1003 - CMN_2028, CMN_7136, CMN_17800 - CMN_17856 and CMN_65011 - CMN_65071</td>
<td>CMN Messages: Other common errors</td>
</tr>
<tr>
<td>114</td>
<td>CNX_53117 - CNX_53119</td>
<td>CNX Messages: Connection-related errors</td>
</tr>
<tr>
<td>115</td>
<td>CONF_45006</td>
<td>CONF Messages: Configuration-related errors</td>
</tr>
<tr>
<td>116</td>
<td>CSE_34005 - CSE_34042</td>
<td>CSE Messages: Integration service errors</td>
</tr>
<tr>
<td>117</td>
<td>CTS DK_43000 - CTS DK_43003</td>
<td>CTS DK Messages: Cannot find data or library error</td>
</tr>
<tr>
<td>118</td>
<td>DBG R_25011 - DBG R_25068</td>
<td>DBG R Messages: Internal or data-related errors</td>
</tr>
<tr>
<td>119</td>
<td>DMI_17501 - DMI_17527</td>
<td>DMI Messages: Metadata-related errors</td>
</tr>
<tr>
<td>130</td>
<td>HIER_28004 - HIER_28051</td>
<td>HIER Messages</td>
</tr>
<tr>
<td>Exit Code</td>
<td>Informatica Error Code Range</td>
<td>Informatica Error Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>131</td>
<td>ICMD_10000 - ICMD_10343</td>
<td>ICMD Messages: Infa command-related errors</td>
</tr>
<tr>
<td>132</td>
<td>IDM_24007 - IDM_26011</td>
<td>IDM Messages: Invalid input or data masking errors</td>
</tr>
<tr>
<td>133</td>
<td>IDOC_17601 - IDOC_17755</td>
<td>IDOC Messages: IDOC related errors</td>
</tr>
<tr>
<td>134</td>
<td>JMS_1001 - JMS_3034</td>
<td>JMS Messages: JNDI or JMS errors</td>
</tr>
<tr>
<td>135</td>
<td>JSDK_42021 - JSDK_42075</td>
<td>JSDK Messages: JVM or Java SDK-related errors</td>
</tr>
<tr>
<td>136</td>
<td>JTX_1001 - JTX_1117 and JTX_60000 - JTX_60014</td>
<td>JTX Messages: Java transformation-related errors</td>
</tr>
<tr>
<td>137</td>
<td>LB_47007 - LB_47050</td>
<td>LB Messages: Load balancer errors</td>
</tr>
<tr>
<td>138</td>
<td>LDBG_8316 and LDBG_21035 - LDBG_21668</td>
<td>LDBG Messages</td>
</tr>
<tr>
<td>139</td>
<td>LGS_10006 - LGS_10060</td>
<td>LGS Messages: Log-related errors</td>
</tr>
<tr>
<td>140</td>
<td>LIC_10000 - LIC_10064</td>
<td>LIC Messages: Licensing errors</td>
</tr>
<tr>
<td>142</td>
<td>LMTGT_17801 - LMTGT_17821</td>
<td>LMTGT Messages: LM API target errors</td>
</tr>
<tr>
<td>143</td>
<td>MBC_EXCL_E001 - MBC_EXCL_E0122</td>
<td>MBC_EXCL Messages: Excel-related errors</td>
</tr>
<tr>
<td>144</td>
<td>MMS_10109</td>
<td>MMS Messages: Metadata manager error</td>
</tr>
<tr>
<td>145</td>
<td>MPSVCCMN_10002</td>
<td>MPSVCCMN Messages: Mapping service error</td>
</tr>
<tr>
<td>146</td>
<td>MQ_29000 - MQ_29280</td>
<td>MQ Messages: Queue manager related errors</td>
</tr>
<tr>
<td>147</td>
<td>NODE_10014</td>
<td>NODE Messages: Node configuration error</td>
</tr>
<tr>
<td>148</td>
<td>NTSERV_10000 - NTSERV_10246</td>
<td>NTSERV Messages: NT service error</td>
</tr>
<tr>
<td>149</td>
<td>NZRDR_10015 - NZRDR_10036</td>
<td>NZRDR Messages</td>
</tr>
<tr>
<td>150</td>
<td>NZWRT_20029 - NZWRT_20126</td>
<td>NZWRT Messages</td>
</tr>
<tr>
<td>Exit Code</td>
<td>Informatica Error Code Range</td>
<td>Informatica Error Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>151</td>
<td>OBJM_54505 - OBJM_54545</td>
<td>OBJM Messages</td>
</tr>
<tr>
<td>152</td>
<td>ODL_26001 - ODL_26116</td>
<td>ODL Messages</td>
</tr>
<tr>
<td>153</td>
<td>OHS_99001 - OHS_99019</td>
<td>OHS Messages: OHS-related errors</td>
</tr>
<tr>
<td>154</td>
<td>OPT_63005 - OPT_63282</td>
<td>OPT Messages</td>
</tr>
<tr>
<td>155</td>
<td>PCCL_97001 - PCCL_97034</td>
<td>PCCL Messages: Connection property-related errors</td>
</tr>
<tr>
<td>156</td>
<td>PCSF_10004 - PCSF_10538 and PCSF_46000 - PCSF_46015</td>
<td>PCSF Messages: Power center related errors</td>
</tr>
<tr>
<td>157</td>
<td>PETL_24042 - PETL_24074</td>
<td>PETL Messages: DTM Errors</td>
</tr>
<tr>
<td>158</td>
<td>PMF_15000 - PMF_15019</td>
<td>PMF Messages: PMF-related errors</td>
</tr>
<tr>
<td>159</td>
<td>PMJVM_42011</td>
<td>PMJVM Messages: DTM failed to close JVM error</td>
</tr>
<tr>
<td>160</td>
<td>PR_18001 - PR_18040</td>
<td>PR Messages: Peoplesoft-related errors</td>
</tr>
<tr>
<td>161</td>
<td>PWX_34000 - PWX_34504</td>
<td>PWX Messages: Power exchange tasks related errors</td>
</tr>
<tr>
<td>162</td>
<td>REGEX_34005 - REGEX_34014</td>
<td>REGEX Messages: Regex- or syntax-related errors</td>
</tr>
<tr>
<td>163</td>
<td>REP_12001 - REP_12999, REP_22674, REP_32000 - REP_32899, REP_51037 - REP_51961, REP_55035 - REP_58224 and REP_61002 - REP_62386</td>
<td>REP Messages: Repository errors</td>
</tr>
<tr>
<td>164</td>
<td>REP_CORE_59046</td>
<td>REP_CORE Messages: Repository high availability message</td>
</tr>
<tr>
<td>165</td>
<td>RFC_17403 - RFC_17439</td>
<td>RFC Messages: RFC-related errors</td>
</tr>
<tr>
<td>166</td>
<td>RR_4004 - RR_4043</td>
<td>RR Messages: Database-related errors</td>
</tr>
<tr>
<td>167</td>
<td>RS_39037 - RS_39209</td>
<td>RS Messages: Repository service error messages</td>
</tr>
<tr>
<td>168</td>
<td>SAPUTIL_99506 - SAPUTIL_99076</td>
<td>SAPUTIL Messages</td>
</tr>
</tbody>
</table>
### Exit Codes for Job Submission

<table>
<thead>
<tr>
<th>Exit Code</th>
<th>Informatica Error Code Range</th>
<th>Informatica Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>169</td>
<td>SDKC_37005 - SDKC_37008</td>
<td>SDKC Messages</td>
</tr>
<tr>
<td>170</td>
<td>SDKS_38005 - SDKS_38605</td>
<td>SDKS Messages</td>
</tr>
<tr>
<td>171</td>
<td>SDKXML_43009</td>
<td>SDKXML Messages: Java transformation failure error</td>
</tr>
<tr>
<td>172</td>
<td>SF_31143 - SF_34160</td>
<td>SF Messages: Salesforce-related errors</td>
</tr>
<tr>
<td>173</td>
<td>SFDC_31101 - SFDC_31208</td>
<td>SFDC Messages</td>
</tr>
<tr>
<td>174</td>
<td>SM_7024 - SM_7219</td>
<td>SM Messages</td>
</tr>
<tr>
<td>175</td>
<td>SORT_40046 - SORT_40424</td>
<td>SORT Messages: Sort library- and memory-related errors</td>
</tr>
<tr>
<td>176</td>
<td>SPC_10027 - SPC_10052</td>
<td>SPC Messages: Start process or start service errors</td>
</tr>
<tr>
<td>177</td>
<td>SQL_50001 - SQL_50072</td>
<td>SQL Messages: SQL-related errors</td>
</tr>
<tr>
<td>178</td>
<td>SR_17000 - SR_17171</td>
<td>SR Messages</td>
</tr>
<tr>
<td>179</td>
<td>TE_7102 - TE_7135</td>
<td>TE Messages</td>
</tr>
<tr>
<td>180</td>
<td>TIB_34001 - TIB_38014</td>
<td>TIB Messages: TIBCO-related error messages</td>
</tr>
<tr>
<td>181</td>
<td>TM_6004 - TM_6958</td>
<td>TM Messages</td>
</tr>
<tr>
<td>182</td>
<td>TT_11009 - TT_11210</td>
<td>TT Messages: Transformation errors</td>
</tr>
<tr>
<td>183</td>
<td>UDT_50000 - UDT_50034</td>
<td>UDT Messages: Unstructured data transformation-related errors</td>
</tr>
<tr>
<td>184</td>
<td>UM_10000 - UM_10099, UM_100100 - UM_100102</td>
<td>UM Messages: User management-related errors</td>
</tr>
<tr>
<td>185</td>
<td>VAR_27001 - VAR_27097</td>
<td>VAR Messages: Variable-related errors</td>
</tr>
<tr>
<td>186</td>
<td>WEBM_1001 - WEBM_3005</td>
<td>WEBM Messages: Web methods-related errors</td>
</tr>
<tr>
<td>187</td>
<td>WRT_8000 - WRT_8443 and WRT_31215</td>
<td>WRT Messages: Writer errors</td>
</tr>
<tr>
<td>Exit Code</td>
<td>Informatica Error Code Range</td>
<td>Informatica Error Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>188</td>
<td>WSC_33021 - WSC_33033, WSC_38001 - WSC_38003 and WSC_42008 - WSC_42022</td>
<td>WSC Messages: Web service-related errors</td>
</tr>
<tr>
<td>189</td>
<td>WSH_501 - WSH_1154 and WSH_95063</td>
<td>WSH Messages: Web services hub-related errors</td>
</tr>
<tr>
<td>190</td>
<td>WSP_33002 - WSP_36010</td>
<td>WSP Messages: Web service provider-related errors</td>
</tr>
<tr>
<td>191</td>
<td>XMLR_82061 - XMLR_82078</td>
<td>XMLR Messages: XML read-related error</td>
</tr>
<tr>
<td>192</td>
<td>XMLW_31001 - XMLW_31211</td>
<td>XMLW Messages: XML write-related errors</td>
</tr>
<tr>
<td>200</td>
<td>DOM_10009 - DOM_10189</td>
<td>DOM_10009 - DOM_10189</td>
</tr>
<tr>
<td>201</td>
<td>DP_90001 - DP_90804</td>
<td>DP Messages: Profiling-related errors</td>
</tr>
<tr>
<td>202</td>
<td>DS_10008 - DS_10059</td>
<td>DS Messages</td>
</tr>
<tr>
<td>203</td>
<td>DSP_20307</td>
<td>DSP Messages: Node information-related error</td>
</tr>
<tr>
<td>205</td>
<td>EXP_19007 - EXP_20011</td>
<td>EXP Messages: Expression or syntax-related errors</td>
</tr>
<tr>
<td>206</td>
<td>EXPFN_34016 - EXPFN_34017</td>
<td>EXPFN Messages: Invalid expression for numbers</td>
</tr>
<tr>
<td>207</td>
<td>FEXP_87001 - FEXP_87026</td>
<td>FEXP Messages: Output file-related errors</td>
</tr>
<tr>
<td>208</td>
<td>FR_3000 - FR_3117</td>
<td>FR Messages</td>
</tr>
<tr>
<td>209</td>
<td>FTP_14002 - FTP_14084</td>
<td>FTP Messages: FTP-related errors</td>
</tr>
</tbody>
</table>
Chapter 3: Running the CLI Using Your Scheduling Manager

You can run the informatica script using any CA scheduling manager product including CA Workload Automation DE, CA Workload Automation AE, CA Workload Automation ESP Edition, and CA Workload Automation CA 7 Edition. You can then track the job to completion using the standard product features for monitoring.

**Note:** For more information about using one of these CA scheduling manager products, see the appropriate CA documentation.

This section contains the following topics:
- Integration with CA Workload Automation DE (see page 47)
- Integration with CA Workload Automation AE (see page 49)
- Integration with CA Workload Automation ESP Edition (see page 52)
- Integration with CA Workload Automation CA 7 Edition (see page 54)

## Integration with CA Workload Automation DE

To run the informatica script using CA Workload Automation DE, you define a UNIX or Windows job in CA WA Desktop Client that runs the script. The job is defined as part of an Application and runs on a UNIX or Windows agent.

**Note:** For more information about defining UNIX or Windows jobs using CA Workload Automation DE, see the CA Workload Automation DE Define Perspective Help.

### Example: List the Workflows that Match a Specified Pattern Using a Windows Job

This example lists the jobs defined in Informatica using a Windows job. The following CLI arguments are passed to the batch script:

- **Command** (listAllWorkflows)
- **Folder name** (IPN)
- **Repository name** (Infa_Rep_Srv)
- **User** (U_JAPI)
- **Password** (32704AC768A31370)
- **Matching pattern** ("%wf%_%")
To list the workflows using a Windows job

1. Create a Windows job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—listAllWorkflows
   - Agent—WINAGENT
   - Command to run—\informatica.bat
   - Arguments to pass—repositoryName=Infa_Rep_Srv --user=U_JAPI --password=32704AC768A31370 --command=listAllWorkflows --folderName=IPN --matchingPattern="%wf%_%"
3. Click OK.

After the job completes, you can view the list of workflows matching the specified pattern in the spool file.

Example: Start a Workflow Using a UNIX Job

This example starts a workflow using a UNIX job. The following CLI arguments are passed to the shell script:
- Command (startWorkflow)
- Folder name (ipn)
- Informatica target (tick)
- User (u_japi)
- Password (32704AC768A31370)
- Workflow name (wf_inst)

To start a workflow using a UNIX job

1. Create a UNIX job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—startWorkflow
   - Agent—LINUIXAGENT
   - Script/command name—wrappers/informatica
   - Arguments to pass—infaTarget=tick --user=u_japi --password=32704AC768A31370 --command=startWorkflow --folderName=ipn --workflowName=wf_inst
3. Select the Run a script option button.
4. Click OK.
Example: Stop a Running Workflow Using a Windows Job

This example stops a running workflow using a Windows job. The following CLI arguments are passed to the batch script:

- Command (stopWorkflow)
- Folder Name (ipn)
- Workflow name (wf_inst)

To stop a running workflow using a Windows job

1. Create a Windows job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—stopWorkflow
   - Agent—WINAGENT
   - Command —rwrappers\informatica.bat
   - Arguments to pass—command=stopWorkflow --folderName=ipn --workflowName=wf_inst
3. Click OK.

Integration with CA Workload Automation AE

To run the informatica script using CA Workload Automation AE, you define a command job in JIL or CA Workload Control Center that runs the script. The job runs on a UNIX or Windows agent.

Notes:

- You cannot insert a job if the value specified in the command exceeds 512 bytes. To avoid this limitation, embed the entire CLI command in a batch or shell script and specify the full path to the script as the command.
- For more information about defining command jobs using CA Workload Automation AE, see the CA Workload Automation AE User Guide and the CA Workload Automation AE Reference Guide.
- For more information about defining command jobs using CA Workload Control Center, see the CA Workload Control Center Workload Scheduling Guide.
Example: List the Workflows that Match a Specified Pattern Using a Command Job

This example lists the workflows that match a specified pattern using a command job. The job runs on a Windows agent. The following CLI arguments are passed to the batch script:

- Command (listAllWorkflow)
- Folder Name (IPN)
- Repository name (Infa_Rep_Srv)
- User (U_JAPI)
- Password (32704AC768A31370)
- Matching pattern ("%wf%_%")

To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```
insert_job: listAllWorkflows
job_type: CMD
machine: winagent
command: wrappers\informatica.bat
    --repositoryName=Infa_Rep_Srv --user=U_JAPI
    --password=32704AC768A31370 --command=listAllWorkflows --foldername=IPN
    --matchingpattern="%wf%_%"
```

To define the command job in CA Workload Control Center

1. Create a Command job.
2. Enter the following properties:
   - Name—listAllWorkflows
   - Send to machine—WINAGENT
   - Command—wrappers\informatica.bat
     --repositoryName=Infa_Rep_Srv --user=U_JAPI
     --password=32704AC768A31370
     --command=listAllWorkflows --folderName=IPN --matchingPattern="%wf%_%"
3. Commit the job.
Example: Start a Workflow Using a Command Job

This example starts a workflow using a command job. The job runs on a Linux agent. The following CLI arguments are passed to the shell script:

- Command (startWorkflow)
- Folder Name (IPN)
- Informatica target (Tick)
- User (U_JAPI)
- Password (32704AC768A31370)
- Workflow name (wf_inst)

To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```plaintext
insert_job: startWorkflow
job_type: CMD
machine: linuxagent
command: wrappers/informatica
  --infaTarget=Tick --user=U_JAPI --password=32704AC768A31370
  --command=startWorkflow
  --folderName=IPN --workflowName=wf_inst
```

To define the command job in CA Workload Control Center

1. Create a Command job.
2. Enter the following properties:
   - Name—run_job
   - Send to machine—LINUXAGENT
   - Command—wrappers/informatica
     --infaTarget=Tick --user=U_JAPI --password=32704AC768A31370
     --command=startWorkflow --folderName=IPN --workflowName=wf_inst
3. Commit the job.

Example: Stop a Running Workflow Using a Command Job

This example stops a running workflow using a command job. The job runs on a Windows agent. The following CLI arguments are passed to the batch script:

- Command (stopWorkflow)
- Folder Name (IPN)
- Workflow name (wf_inst)
To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```plaintext
insert_job: stopWorkflow
job_type: CMD
machine: winagent
command: wrappers\informatica.bat --command=stopWorkflow --folderName=IPN --workflowName=wf_inst
```

To define the command job in CA Workload Control Center

1. Create a Command job.
2. Enter the following properties:
   - Name—run_job
   - Send to machine—WINAGENT
   - Command—wrappers\informatica.bat --command=stopWorkflow --folderName=IPN --workflowName=wf_inst
3. Commit the job.

Integration with CA Workload Automation ESP Edition

To run the informatica script using CA Workload Automation ESP Edition, you define a UNIX or Windows job that runs the script. The job is defined as part of an Application and runs on a UNIX or Windows agent.

Note: For more information about defining UNIX or Windows jobs using CA Workload Automation ESP Edition, see the CA Workload Automation Agent for UNIX, Linux, or Windows User Guide.

Example: List the Workflows that Match a Specified Pattern Using a Windows Job

This example lists the workflows that match a specified pattern using a Windows job. The following CLI arguments are passed to the batch script:

- Command (listAllWorkflows)
- Folder name (IPN)
- Repository name (Infa_Rep_Srv)
- User (U_JAPI)
- Password (32704AC768A31370)
- Matching pattern ("%wf%_%")
The following job definition lists the workflows that match a specified pattern using a Windows job:

```
NT_JOB LISTALLWORKFLOWS
    AGENT WINAGENT
    CMDNAME wrappers\informatica.bat
    ARGS --repositoryName=Infa_Rep_Srv --user=U_JAPI --password=32704AC768A31370 +
    --command=listAllWorkflows --folderName=IPN --matchingPattern="%wf%_%"
    RUN DAILY
ENDJOB
```

**Example: Start a Workflow Using a UNIX Job**

This example starts a workflow using a UNIX job. The following CLI arguments are passed to the shell script:

- Command (startWorkflow)
- Folder name (IPN)
- Informatica target (tick)
- User (u_japi)
- Password (32704AC768A31370)
- Workflow name (wf_inst)

The following job definition starts a workflow using a UNIX job:

```
UNIX_JOB STARTWORKFLOW
    AGENT LINUXAGENT
    CMDNAME wrappers/informatica
    ARGS infaTarget=Tick --user=U_JAPI --password=32704AC768A31370 +
    --command=startWorkflow --folderName=IPN --workflowName=wf_inst
    RUN DAILY
ENDJOB
```

**Example: Stop a Running Workflow Using a Windows Job**

This example stops a running workflow using a Windows job. The following CLI arguments are passed to the batch script:

- Command (stopWorkflow)
- Folder Name (IPN)
- Workflow name (wf_inst)
The following job definition stops a running workflow using a Windows job:

```
NT_JOB STOPWORKFLOW
    AGENT WINAGENT
    CMDNAME wrappers\informatica.bat
    ARGSS --command=stopWorkflow --folderName=IPN --workflowName=wf_inst
    RUN DAILY
ENDJOB
```

**Integration with CA Workload Automation CA 7 Edition**

To run the informatica script using CA Workload Automation CA 7 Edition, you define a UNIX or Windows job that runs the script. The job runs on a UNIX or Windows agent.

**Note:** For more information about defining UNIX or Windows jobs using CA Workload Automation CA 7 Edition, see the *CA Integrated Agent Services User Guide*.

**Example: List the Workflows that Match a Specified Pattern Using a Windows Job**

This example lists the workflows that match a specified pattern using a Windows job. The following CLI arguments are passed to the batch script:

- Command (listAllWorkflows)
- Folder name (IPN)
- Repository name (Infa_Rep_Srv)
- User (U_JAPI)
- Password (32704AC768A31370)
- Matching pattern ("%wf%_%")

The following job definition lists the workflows that match a specified pattern using a Windows job:

```
AGENT WINAGENT
    CMDNAME wrappers\informatica.bat
    ARGSS --repositoryName=Infa_Rep_Srv --user=U_JAPI --password=32704AC768A31370 +
           --command=listAllWorkflows --folderName=IPN --matchingPattern="%wf%_%"
```
Example: Start a Workflow Using a UNIX Job

This example starts a workflow using a UNIX job. The following CLI arguments are passed to the shell script:

- Command (startWorkflow)
- Folder name (IPN)
- Informatica target (tick)
- User (u_japi)
- Password (32704AC768A31370)
- Workflow name (wf_inst)

The following job definition starts a workflow using a UNIX job:

AGENT LINUXAGENT
CMDNAME wrappers/informatica
ARGS infaTarget=tick --user=U_JAPI --password=32704AC768A31370 + --command=startWorkflow --folderName=IPN --workflowName=wf_inst

Example: Stop a Running Workflow Using a Windows Job

This example stops a running workflow using a Windows job. The following CLI arguments are passed to the batch script:

- Command (stopWorkflow)
- Folder Name (IPN)
- Workflow name (wf_inst)

The following job definition stops a workflow using a Windows job:

AGENT WINAGENT
CMDNAME wrappers\informatica.bat
ARGS --command=stopWorkflow --folderName=IPN --workflowName=wf_inst