IBM Workload Scheduler

# Scenarios and How To Demos

*Version 9 Release 4*

IBM Workload Scheduler

# Scenarios and How To Demos

*Version 9 Release 4*

**Note**
Before using this information and the product it supports, read the information in "Notices" on page 51.

This edition applies to version 9, release 4, modification level 0 of IBM Workload Scheduler (program number 5698-WSH) and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# Chapter 1. Scenarios

Scenarios demonstrating practical application of IBM Workload Scheduler in a business environment.

View these scenarios to help you get familiar with IBM Workload Scheduler and learn how to use the product to achieve your business goals.

You can find additional scenarios at the following links:

- The Workload Automation YouTube channel, which is continuously updated with video demos that show new features and capabilities for both IBM Workload Scheduler and Workload Automation on Cloud.
- The IBM Workload Scheduler Wiki Media Gallery, which contains demos (only available in English) about how to use IBM Workload Scheduler.
- A workload service assurance scenario in the IBM Workload Scheduler User's Guide and Reference, which describes how to monitor critical jobs.

# Chapter 2. Dynamic job scheduling scenarios

This section describes the dynamic job scheduling scenarios.

## Setting up your environment for dynamic scheduling

This section describes the prerequisite tasks you perform before running the example dynamic scheduling scenarios in your IBM Workload Scheduler environment.

- If you are installing IBM Workload Scheduler for the first time:
  1. Install a master domain manager using the procedures described in *IBM Workload Scheduler: Planning and Installation*. Tick the check box option to create the Dynamic Workload Broker CPU definition and activate the dynamic scheduling capability. After the installation completes, the broker application is started automatically.
  2. Install the IBM Workload Scheduler agents by selecting the **Add Dynamic scheduling capabilities** during the installation.
  3. Install the Dynamic Workload Console

For a step by step installation or upgrade procedure, see *IBM Workload Scheduler: Planning and Installation*

For further details about configuring your environment for dynamic scheduling, see *IBM Workload Scheduler: Administration Guide*.

## Applying an order of preference to possible targets

In this scenario, you create a job that uses logical resources to apply an order of preference to a group of eligible target computers on which a job can run.

### Business Goal

A company needs to establish an order of preference when selecting a target computer to run a job. The job is sent to the first available computer based on the specified order of preference policy. This is useful when more than one computer has suitable characteristics to run a given job. New machines can be added to the preference order and if two or more machines match the needed criteria, the order of preference is applied to select the target. In this scenario, target computers have similar or identical characteristics and are typically used as backups for each other during scheduled maintenance or in case of failover. The scenario consists of the following tasks:

1. The administrator creates a logical resource for each of the eligible computer targets that are part of the preference order list. The logical resources are all of the same type and have a quantity that represents the preference value used to order the list.
2. The administrator creates a definition for the job that must be run on the target computers according to the preference order.
3. The administrator uses the Dynamic Workload Console to schedule the job.

### Roles

This section lists the user roles needed to run the scenario:

**dynamic workload broker Developer**
Defines the jobs using the Job Brokering Definition Console

**dynamic workload broker Operator**
Monitors and controls the jobs that have been submitted.

**IBM Workload Scheduler Job Scheduler**
Manages the workload by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- A IBM Workload Scheduler network with the Dynamic Scheduling capability.
- Optionally, the Dynamic Workload Console

### Setting up the environment

To run this scenario, complete the tasks described in "Setting up your environment for dynamic scheduling" on page 3.

## Running the Scenario

To create the logical resources and a job definition that selects a defined logical resource based on an established order of preference, perform the following steps:

### Procedure

1. Select the **Define a New Logical Resource** task option in the Dynamic Workload Consoleto create the logical resources needed for the job. The job must run on a Linux workstation. ComputerA, ComputerB, and ComputerC are Linux workstations that are suitable for running the job. ComputerA is the first preference. ComputerB is considered to be a backup if ComputerA is unavailable, and ComputerC is used if neither ComputerA nor ComputerB is available. Define the logical resources with the following characteristics:

*Table 1. Logical resources for defining an order of preference*

| Logical resource name | Type | Quantity | Computer |
|---|---|---|---|
| Preference1 | Preference | 50 | ComputerA |
| Preference2 | Preference | 30 | ComputerB |
| Preference3 | Preference | 10 | ComputerC |

2. To create the job definition, in the Job Brokering Definition Console select **File** > **New** > **Job brokering definition** and create a new job definition named `jobWithPreference`. The job definition opens at the Overview page with the job name assigned.
3. Open the Application page and define the required information for the application that the job is to run. In the **Type** field select **Executable**. Specify the executable name and path. In this definition, the variable `myApp` is used to identify the executable `myExecutable` as shown in the sample JSDL file.
4. Open the Resources page and select the Software Requirements tab.

5. Create an operating system requirement, as follows:
   a. In the Candidate Operating Systems pane, click **Add**. The Operating System Details dialog box is displayed.
   b. In the **Type** field, select **LINUX** and click **OK**.
6. Create a requirement for a logical resource, as follows:
   a. In the Logical Resources pane, click **Add**. The Logical Resource Details dialog box is displayed.
   b. In the **Type** field, type `Preference` (the type assigned to the three logical resources) and click **OK**.
7. Open the Optimization page and specify an optimization objective, as follows:
   a. In the **Type** menu, choose **Select best resource by optimization objective**.
   b. In the **Resource Type** menu, select **Logical Resource**.
   c. In the **Resource Property** menu, select **Quantity**.
   d. In the **Optimization Objective** menu, select **Maximize**.

   When the job is submitted, it is sent to the first available workstation associated with the logical resource of type **Preference** that has the highest quantity available.
8. Select **File** > **Save** to save the job definition file.
9. Select the new JSDL and upload it to the server by clicking on the corresponding icon.
10. Submit the job in one of the following ways, depending on whether you want to submit it as a broker job or by using a job definition.
    - Log in to the Dynamic Workload Console and choose the **Dynamic Workload Broker** portfolio option.
      – Select **Definitions** > **Jobs** . Optionally specify search criteria and click **Search**. Select the job definition you created in the previous steps.
      – To run the job, select **Submit** and click **Go**.
    - Log in to the Dynamic Workload Console.
      – Select **Administration** > **Workload Design** > **Manage Workload Definitions**to open the Workload Designer. Create a new broker job definition by filling in the required fields as appropriate. In the **Workload Broker Job Name** field, type the name of the JSDL file that you created in the previous steps.
      – Submit the IBM Workload Scheduler job by selecting **Workload** > **Submit** > **Submit Predefined Jobs** .

    You can also add the job to an existing job stream , or submit the job using the **jobsubmit** command. For further information about the dynamic workload broker command-line interface, see *IBM Workload Scheduler: Scheduling Workload Dynamically*.

## Expected Results

When the job is submitted, it is sent to the first available target computer associated with the logical resource of type **Preference** that has the highest quantity. If a selected target is unavailable, the job is submitted to the first available target in the order of preference. You can change preference values for your computers at any time. The new order becomes effective for job submission as soon as it is saved.

## Sample Configuration file

The JSDL file created for this scenario has the following syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
  xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle"
     name="JobWithPreference">
  <jsdl:variables>
    <jsdl:stringVariable name="myApp">myExecutable</jsdl:stringVariable>
  </jsdl:variables>
  <jsdl:application name="executable">
    <jsdle:executable interactive="false" path="${myApp}"/>
  </jsdl:application>
  <jsdl:resources>
    <jsdl:candidateOperatingSystems>
      <jsdl:operatingSystem type="LINUX"/>
    </jsdl:candidateOperatingSystems>
    <jsdl:logicalResource subType="Preference"/>
  </jsdl:resources>
  <jsdl:optimization name="JPT_BestResource">
    <jsdl:objective propertyObjective="maximize" resourcePropertyName="Quantity"
      resourceType="LogicalResource"/>
  </jsdl:optimization>
</jsdl:jobDefinition>
```

# Performing load balancing based on available free memory

In this scenario, an optimization policy is defined to balance the distribution of a set of jobs between resources based on total physical memory available on each resource.

## Business Goal

A company wants to set up job definitions for a group of jobs that must run at about the same time on similar AIX® computers. To ensure that the jobs are efficiently distributed between available resources, each job definition includes an optimization objective to maximize the free physical memory. When the jobs are submitted, a policy is applied to distribute the jobs between the available resources according to the amount of free physical memory on each resource. All matching resources are considered and the load is balanced proportionally between them, using a policy that allocates more jobs to those resources having higher amounts of free physical memory.

This policy is applicable when a set of jobs are clearly affected by a specific type of resource (CPU usage, available memory, available disk space) and are to be run with a high degree of parallelism.

- The administrator has a set of jobs that must be run on AIX machines and that have specific physical memory requirements. This requires that the job is sent to a machine with low physical memory utilization. Several computers are available, each with an equivalent software configuration but with differences in hardware characteristics and actual load. He creates a job definition for each of the jobs and specifies that the job must run on AIX, has an objective of type **Balance load between resources by optimization objective** and should be optimized with the policy **Maximize FreePhysicalMemory**.
- The administrator uses theDynamic Workload Console to submit or schedule the jobs for submission.

### Roles

This section lists the user roles needed to run the scenario:

**dynamic workload broker Developer**
Defines the jobs using the Job Brokering Definition Console

**dynamic workload broker Operator**
Monitors and controls the jobs that have been submitted.

**IBM Workload Scheduler Job Scheduler**
Manages Tivoli® Workload Scheduler workload by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- A IBM Workload Scheduler version 8.5.1 network with the dynamic scheduling capability.
- Optionally, Dynamic Workload Console version 8.5.1.

### Setting up the environment

To run this scenario you must install or upgrade to IBM Workload Scheduler version 8.5.1. Complete the tasks described in "Setting up your environment for dynamic scheduling" on page 3.

## Running the Scenario

To create a job definition that uses optimization policies, perform the following steps:

### Procedure

1. In the Job Brokering Definition Console select **File** > **New** > **Job brokering definition** and create a new job definition named `jobBalancedbyAvailRAM`. The job definition opens at the Overview page with the job name assigned.
2. Open the Application page and define the required information for the application that the job is to run. In the **Type** field select **Executable**. In this definition, the variable `myApp` is used to identify the executable `myExecutable` as shown in the sample JSDL file.
3. Open the Resources page and select the Software Requirements tab.
4. Create an operating system requirement, as follows:
   a. In the Candidate Operating Systems pane, click **Add**. The Operating System Details dialog box is displayed.
   b. In the **Type** field, select **AIX** and click **OK**.
5. Open the Optimization page and specify an optimization objective, as follows:
   a. In the **Type** menu, select **Balance load between resources by optimization objective**.
   b. In the **Resource Type** menu, select **Operating System**.
   c. In the **Resource Property** menu, select **Free Physical Memory**.
   d. In the **Optimization Objective** menu, select **Maximize**.
6. Select **File** > **Save** to save the job definition file.
7. Select the new JSDL and upload it to the server by clicking on the corresponding icon.

8. Submit the job proceeding in one of the following ways, depending on whether you want to submit it as a broker job or by using a Tivoli Workload Scheduler job definition.
   - Log in to the Dynamic Workload Console and choose the **Tivoli Dynamic Workload Broker** portfolio option.
     - Select **Definitions** > **Jobs** . Optionally specify search criteria and click **Search**. Select the job definition that you created in the previous steps.
     - To run the job, select **Submit** and click **Go**.
   - Log in to the Dynamic Workload Console and choose the **Tivoli Workload Scheduler** portfolio option.
     - Select **Workload** > **Design** > **Create Workload Definitions**. In the Workload Designer, create a new broker job definition filling in the required fields as appropriate. In the **Workload Broker Job Name** field, type the name of the JSDL file you created in the previous steps.
     - Submit the IBM Workload Scheduler job by selecting **Workload** > **Submit** > **Submit Predefined Jobs**.

   You can also add the job to an existing job stream , or submit the job using the **jobsubmit** command. For further information about the dynamic workload broker command-line interface, see *IBM Workload Scheduler: Scheduling Workload Dynamically*.

## Expected Results

When the jobs are submitted, they are sent to the available computers in proportion to the amount of available free physical memory. If, for example, the snapshot of the currently available free physical memory is the following:

*Table 2. Available Free Physical memory on eligible target AIX computers*

| Computer A | Computer B | Computer C |
|---|---|---|
| 2048 Mbytes | 3072 Mbytes | 512 Mbytes |

and 11 jobs are submitted within a time slot, the following proportional distribution can occur:

*Table 3. Job distribution on targets*

| Computer A | Computer B | Computer C |
|---|---|---|
| 4 jobs | 6 jobs | 1 job |

## Sample Configuration file

The JSDL file created for this scenario has the following syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
  xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle"
   name="jobBalancedbyAvailRAM">
  <jsdl:variables>
    <jsdl:stringVariable name="myApp">sleep</jsdl:stringVariable>
  </jsdl:variables>
  <jsdl:application name="executable">
    <jsdle:executable interactive="false" path="${myApp}">
      <jsdle:arguments>
        <jsdle:value>30</jsdle:value>
      </jsdle:arguments>
    </jsdle:executable>
  </jsdl:application>
  <jsdl:resources>
    <jsdl:candidateOperatingSystems>
```

```
      <jsdl:operatingSystem type="AIX"/>
    </jsdl:candidateOperatingSystems>
  </jsdl:resources>
  <jsdl:optimization name="JPT_JSDLOptimizationPolicyType">
    <jsdl:objective propertyObjective="maximize"
      resourcePropertyName="FreePhysicalMemory" resourceType="OperatingSystem"/>
  </jsdl:optimization>
</jsdl:jobDefinition>
```

# Specifying software license requirements by using resources

In this scenario, a job requires two software licenses to run. Using logical resources to represent license availability, a company ensures compliance to license requirements.

## Business Goal

A company runs a job that uses two licensed software products. Product SoftwareA is licensed under a node-locked license agreement, for which the company has four entitlements assigned to ComputerA, ComputerB, ComputerC, and ComputerD. For product SoftwareB, the company has a floating license, for which there are three entitlements available to all computers in the network. The job is submitted ensuring compliance to the licensing policies.

### Roles

This section lists the user roles needed to run the scenario:

**dynamic workload broker Developer**
: Defines the jobs using the Job Brokering Definition Console

**dynamic workload broker Operator**
: Monitors and controls the jobs that have been submitted.

**IBM Workload Scheduler Job Scheduler**
: Manages the workload by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- A IBM Workload Scheduler network with the dynamic scheduling capability.
- Optionally, the Dynamic Workload Console.

### Setting up the environment

To run this scenario, complete the tasks described in “Setting up your environment for dynamic scheduling” on page 3.

## Running the Scenario

### Before you begin

Before creating the job, the administrator uses the **Define a New Logical Resource Task** option in the Dynamic Workload Console to create the logical resources, as shown in Table 4 on page 10. The node-locked license logical resources are each assigned to a computer and the quantities specified can only be used by jobs

running on the specified computer. The floating license logical resource is a global resource. It is not assigned to a computer and can be accessed by a job running on any computer until the total quantity is used.

*Table 4. Logical resources for controlling license availability*

| logical resource name | Type | Quantity | Computer |
|---|---|---|---|
| locked_license1 | SoftwareA | 1 | ComputerA |
| locked_license2 | SoftwareA | 1 | ComputerB |
| locked_license3 | SoftwareA | 1 | ComputerC |
| locked_license_4 | SoftwareA | 1 | ComputerD |
| floating_license | SoftwareB | 3 | None |

To create a job definition that uses these logical resources to include license requirements, perform the following steps:

## Procedure

1. In the Job Brokering Definition Console, select **File** > **New** > **Job Brokering Definition** and create a new job definition named `jobThatConsumesLicenses`. The job definition opens at the Overview page with the job name assigned.
2. Open the Application page and define the required information for the application that the job is to run.
3. Open the Resources page and select the Software Requirements tab.
4. Create a requirement for a SoftwareA logical resource, as follows:
   a. In the Logical Resources pane, click **Add**. The Logical Resource Details dialog box is displayed.
   b. In the **Type** field, type `SoftwareA` and click **OK** . The Logical Resource details are displayed.
   c. In the Logical Resource details area, specify 1 in the quantity field.
5. Open the Related Resource page and create a requirement for the SoftwareB logical resource, as follows:
   a. In the Resource Requirements pane, click **Add**. The Resource Requirement Details dialog box is displayed.
   b. In the **ID** field, specify a meaningful ID, in this example, `Floating_Global_License`.
   c. In the **Type** field, select **Logical Resource** and click **OK**.
   d. In the Resource Properties pane, click **Add Requirement**. The Resource Requirement Details dialog box is displayed.
   e. Select **Subtype** from the **Property name** menu, type `SoftwareB` in the **Property Value** field, and click **OK**.
   f. In the Allocations pane, click **Add**. The Allocations Details dialog box is displayed.
   g. Select **Quantity** from the **Property name** menu, type 1 in the **Quantity** field, and click **OK**.
6. Select **File** > **Save** to save the job definition file.
7. Select the new JSDL and upload it to the server by clicking on the corresponding icon.
8. Submit the job proceeding in one of the following ways, depending on whether you want to submit it as a broker job or using a job definition.

- Log in to the Dynamic Workload Console and choose the **Dynamic Workload Broker** portfolio option.
  - Select **Definitions** > **Jobs** Optionally specify search criteria and click **Search**. Select the job definition that you created in the previous steps.
  - To run the job, select **Submit** and click **Go**.
- Log in to the Dynamic Workload Console.
  - Select **Administration** > **Workload Design** > **Manage Workload Definitions**. In the Workload Designer, create a new broker job definition filling in the required fields as appropriate. In the **Workload Broker Job Name** field, type the name of the JSDL file you created in steps 1 through 7.
  - Submit the job by selecting **Administration** > **Workload Submission** > **Submit Predefined Jobs** .

  You can also add the job to an existing job stream , or submit the job using the **jobsubmit** command. For further information about the dynamic workload broker command-line interface, see *IBM Workload Scheduler: Scheduling Workload Dynamically*.

## Expected Results

When the jobs are submitted, they are sent to one of the nodes having a node-locked license and using one of the floating global licenses, in accordance with the licensing policies. With the settings used in this scenario, the maximum number of `jobthatConsumesLicenses` instances that can run concurrently is 3 because each of them are allocated a floating license of `SoftwareB`. The instances can either run on the same or on different computers holding the node-locked license of `SoftwareA`.

## Sample Configuration file

The JSDL file created for this scenario has the following syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
  xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle"
    name="jobThatConsumesLicences">
  <jsdl:variables>
    <jsdl:stringVariable name="myApp">sleep</jsdl:stringVariable>
  </jsdl:variables>
  <jsdl:application name="executable">
    <jsdle:executable interactive="false" path="${myApp}">
      <jsdle:arguments>
        <jsdle:value>120</jsdle:value>
      </jsdle:arguments>
    </jsdle:executable>
  </jsdl:application>
  <jsdl:resources>
    <jsdl:logicalResource quantity="1" subType="SoftwareA"/>
  </jsdl:resources>
  <jsdl:relatedResources id="Floating_Global_License" type="LogicalResource">
    <jsdl:properties>
      <jsdl:requirement propertyName="SubType">
        <jsdl:exact>SoftwareB</jsdl:exact>
      </jsdl:requirement>
    </jsdl:properties>
    <jsdl:allocation propertyName="Quantity">1.0</jsdl:allocation>
  </jsdl:relatedResources>
</jsdl:jobDefinition>
```

# Dynamic job scheduling - A procedure to optimize writing job definitions that simulate the use of templates

This scenario shows how you can set up JSDL job definition templates that you can reuse for several job definitions. In this way you can optimize the number of steps required to submit jobs to dynamic workload broker. Taking this concept a little further, you can set up a different JSDL template for every class of job that you want to submit. For background information, see *Using JSDL job definition templates* in *Scheduling workload dynamically*.

## Scenario goal

Typically, to run a job using dynamic scheduling, you write two definitions for the job: the definition for the database (that you write with composer or enter in the Dynamic Workload Console), and one definition stored in the Job Repository of dynamic workload broker (for which you use the Job Brokering Definition Console). The first definition specifies the type of job, the task string or the name of the executable, the name of the workstation where the job is to run (which for dynamic scheduling must be the workload broker workstation), scheduling dates, and recovery options. The second definition specifies the task string or the name of the executable, the types and quantities of resources needed to run the job, and optimization and scheduling details. There must be a 1 to 1 match between the two definitions. This means that you are duplicating work for every job that you want to submit by using dynamic workload broker.

You can avoid part of this work by following a number of steps to write a single JSDL definition file in the Job Brokering Definition Console that can be referenced by several job definitions. You can then use this file as a template to which you can link all the job definitions that match the required resources, and optimization and scheduling preferences specified in the template.

### Roles

This section lists the user roles needed to run the scenario:

**dynamic workload broker Developer**
Defines the jobs using the Job Brokering Definition Console

**dynamic workload broker Operator**
Monitors and controls the jobs that have been submitted.

**IBM Workload Scheduler Job Scheduler**
Manages the workload by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- IBM Workload Scheduler with the dynamic workload broker feature enabled
- Optionally, the Dynamic Workload Console.

## Running the Scenario

To complete the scenario, perform the following steps:

## Procedure

1. In the Job Brokering Definition Console, create a JSDL document and give it a name. In the Application page, after setting the `Type` to `Executable`, specify the following variable name in the `Script` field of the executable: `${tws.job.taskstring}`. Fill in the data in the remaining pages, specifying the required resources, and the optimization and scheduling details. Save the document in the database. This is your template.
2. With `composer` or with the Dynamic Workload Console define a workstation of type `extended agent` hosted by the workload broker workstation.

   If you need background information about extended agents, see the *IBM Workload Scheduler: User's Guide and Reference*. To create the template, you only need to know the following facts about an extended agent:

   - It is a logical definition that must be hosted by a physical workstation. For JSDL templates, the physical workstation must always be the workload broker workstation. The workload broker workstation is automatically installed when you install IBM Workload Scheduler with dynamic scheduling capabilities. This workstation can host as many extended agents as you need.
   - It requires an access method. An access method can be a complex program, but in this case it is only a statement that references the name of the JSDL document that is your template.
3. Using `composer` or the Dynamic Workload Console write the job definitions that you want to match with the new template. To link these job definitions with your template, write the name of the extended agent as the workstation where the jobs are to run.
4. Add the extended agent to the plan as you do for any other workstation. The workload broker workstation has the task of managing the lifecycle of the extended agent, notifying the master domain manager that it is up and running.

## Expected Results

When jobs are run on the extended agent, they are routed to the workload broker workstation, which handles them differently from other jobs. Instead of searching for the name of the JSDL definition in the task string of the job, the workload broker workstation:

1. Gets the name of the target JSDL from the access method, and passes the task string as a value for variable `${tws.job.taskstring}`.
2. Replaces the task string value in the script element of the target JSDL, and is used as a command string to run on the target agent that is dynamically selected by the dynamic workload broker.

   The JSDL definition invoked by the workload broker workstation works as a sort of template that you can use to run different task strings defined in different IBM Workload Scheduler jobs: the same JSDL document is reused for multiple jobs.

## Sample Configuration files

You want to use dynamic workload broker to run three jobs that share the same resource requirements and optimization and scheduling preferences. You want to use a JSDL template to minimize the required number of definitions. The jobs are named `SUBMIT_JOBXA1`, `SUBMIT_JOBXA2`, and `SUBMIT_JOBXA3`. The following definitions achieve this:

- The definition of the workload broker workstation. It is named `DGCENTER_DWB` and it is of type `BROKER`. There can be only one workload broker workstation

running at a time in an IBM Workload Scheduler network (this applies also to the dynamic workload broker active instance).

```
CPUNAME DGCENTER_DWB
  OS OTHER
  NODE DGCENTER TCPADDR 41111
  ENGINEADDR 31111
  DOMAIN MASTERDM
  FOR MAESTRO
    TYPE BROKER
    AUTOLINK ON
    BEHINDFIREWALL OFF
    FULLSTATUS OFF
END
```

- The definition of extended agent DGCENTER_DWBXA. The extended agent must:
  - Be hosted by the workload broker workstation (DGCENTER_DWB in this example).
  - Include the access method. While normally the ACCESS keyword is followed by the name of the program that implements the specific access method, for JSDL templates it needs only to define the name of the JSDL document that you use as the template - that must be available in a local directory in the workstation where you run the Job Brokering Definition Console - and whatever other parameters you want to use. These items must be enclosed between double quotes.

    This requires that you created the JSDL document you will be using as a template (named SJT in this example), defining the required resources, candidate hosts, and scheduling and optimization preferences, and specifying ${tws.job.taskstring} in the Script field of the executable.

```
CPUNAME DGCENTER_DWBXA
  OS OTHER
  NODE DGCENTER TCPADDR 41111
  FOR MAESTRO HOST DGCENTER_DWB ACCESS "/jsdl/SJT -var target=D:\vmware,RES=RES1"
    TYPE X-AGENT
    AUTOLINK OFF
    BEHINDFIREWALL OFF
    FULLSTATUS OFF
END
```

- The job definitions in IBM Workload Scheduler.
  - The definition of job SUBMIT_JOBXA1:

```
DGCENTER_DWBXA#SUBMIT_JOBXA1
 SCRIPTNAME "C:\Utils\Jobs\javacount_on.bat"
 STREAMLOGON Administrator
 DESCRIPTION "Counts files in APPMX directory."
 TASKTYPE WINDOWS
 RECOVERY RERUN
```

  - The definition of job SUBMIT_JOBXA2:

```
DGCENTER_DWBXA#SUBMIT_JOBXA2
 SCRIPTNAME "D:\Callcenter\Tasks\sortccalls.bat"
 STREAMLOGON Administrator
 DESCRIPTION "Sorts calls by customer."
 TASKTYPE WINDOWS
 RECOVERY RERUN
```

  - The definition of job SUBMIT_JOBXA3:

```
DGCENTER_DWBXA#SUBMIT_JOBXA3
 SCRIPTNAME "C:\Sales\Tools\runstats.bat"
 STREAMLOGON Administrator
 DESCRIPTION "Calculations on totalled business."
 TASKTYPE WINDOWS
 RECOVERY RERUN
```

Because the jobs are defined to run on extended agent DGCENTER_DWBXA, hosted by the workload broker workstation and matched with the SJT JSDL definition, the process:

1. Submits the jobs via dynamic workload broker
2. Uses the specifications of the SJT JSDL definition
3. Replaces variable ${tws.job.taskstring} with the SCRIPTNAME of each job when the job is submitted.

# Scheduling jobs dynamically on SAP R/3 systems - A high availability scenario

This scenario shows how to achieve high availability when scheduling critical jobs dynamically on SAP R/3 systems. Tivoli Workload Scheduler for Applications users can take advantage of the dynamic scheduling capabilities of IBM Workload Scheduler version 8.5.1 by defining a group of two or more IBM Workload Scheduler agents, configured to schedule on a given SAP R/3 system, as completely interchangeable. If an agent is unavailable to schedule on the SAP R/3 system, another agent in the group is dynamically selected.

## Business goal

Tivoli Workload Scheduler for Applications users scheduling on SAP R/3 systems can achieve 24 x 7 scheduling services on their vendor-acquired software by implementing the dynamic scheduling scenario described in this section.

### Roles

This section lists the user roles needed to run the scenario:

**dynamic workload broker Developer**
Defines the jobs using the Job Brokering Definition Console

**dynamic workload broker Operator**
Monitors and controls the jobs that have been submitted.

**IBM Workload Scheduler Job Scheduler**
Manages Tivoli Workload Scheduler workload by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- A IBM Workload Scheduler version 8.5.1 network with the dynamic scheduling capability.
- Tivoli Workload Scheduler for Applications version 8.5 or earlier
- Dynamic Workload Console version 8.5.1.

### Setting up the environment

Complete the following tasks before running the scenario:

- If you are already using Tivoli Workload Scheduler Version 8.5 or earlier and Tivoli Workload Scheduler for Applications for SAP R/3 (r3batch method):
  1. Upgrade your Tivoli Workload Scheduler master domain manager installation to version 8.5.1 using the procedures described in *IBM Workload*

*Scheduler: Planning and Installation*. During the upgrade procedure, tick the checkbox option to create the Dynamic Workload Broker CPU definition and activate the dynamic scheduling capability After the upgrade completes, the broker application is started automatically. If you do not choose this option, after the upgrade completes you must start the Broker application manually by running the `StartBrokerApplication.bat` from the wastools directory on your master domain manager.

**Note:** If in your network you have a backup master domain manager of a previous release, you must install the latest fix pack level before you launch the upgrade. The latest fix pack levels are:

- For IBM Workload Scheduler version 8.3: fix pack 8.
- IBM Workload Scheduler version 8.4: fix pack 4.

2. Upgrade the agents you intend to use for dynamic scheduling to IBM Workload Scheduler version 8.5.1. These are the agents on which Tivoli Workload Scheduler for Applications is also installed. During the upgrade procedure, select **Add Dynamic scheduling capability**. This feature installs the IBM Workload Scheduler agent and starts it.
3. Optionally upgrade the agents you selected to Tivoli Workload Scheduler for Applications version 8.5.
4. On the agent systems which you selected as eligible targets for dynamic scheduling on the SAP R/3 system, choose a common name for the options files. All options files on each of the agents you selected must have the same name.

- If you are installing Tivoli Workload Scheduler Version 8.5.1 for the first time:
  1. Install a IBM Workload Scheduler version 8.5.1 master domain manager using the procedures described in *IBM Workload Scheduler: Planning and Installation*. Tick the checkbox option to create the Dynamic Workload Broker CPU definition and activate the dynamic scheduling capability. After the installation completes, the broker application is started automatically.
  2. Install two or more IBM Workload Scheduler agents by selecting the **Add Dynamic scheduling capability** during the installation. These are the agents that are eligible for dynamic scheduling on your SAP R/3 system.
  3. Install Tivoli Workload Scheduler for Applications version 8.5 on the agents you selected in the previous step.
  4. On the agent systems, choose a common name for the options files. All options files on each of the agents you selected must have the same name.

## Running the Scenario

To complete the scenario, perform the following steps:

### Procedure

1. Log in to the Dynamic Workload Broker Console, and select **Scheduling Environment**.
2. To define the agent computers that are part of the group eligible for scheduling on the remote SAP system, select **Define New Resource Group** to start the wizard.
3. In the Group Type Selection pane, specify the following:
   - In the **Name** field specify the name of the group you are creating.
   - Select the **Computers** radio button and click **Next**.

- In the **Computer Search Criteria** pane specify the search criteria of your choice to select the computers for the resource group and click **Next**.
- Select the computers to include in the resource group from the search results table and click **Next**.
- In the **Summary** pane, check your choices and click **Finish** to create the resource group.

The agent computers belonging to the resource group must each have an options file with the same name.

As an alternative, you can create a logical resource and associate it to the agents you selected.

4. In the Job Brokering Definition Console, select **File** > **New** > **Job brokering definition** to create a new definition for the job that runs on the SAP R/3 system. Specify a name for the definition and click **Finish** to display the Overview page.
5. Open the Application page and type the required information for the job that you are defining:
   a. In the **Type** field, select **Extended Job**.
   b. In the **Task String** field, specify the job instructions to be run on the target SAP R/3 system. The submit job task is the only task supported in this configuration. For example:`-job DYN_SUB -C A -flag type=exec -user TWS4APPS1 -s1 type=A -s1 program=Z_TWS_SLEEP -v1 ONE`.
   c. In the **Task Type** field, specify `LJ` (launch job).
   d. In the **Target** field specify the name of the configuration (options) file which is located on the systems that you defined in the resource group. You can also define and use a variable with a default value for this field. In this way, you can use the same JSDL to schedule on different SAP R/3 systems.
   e. In the **Access Method** field, specify `r3batch.exe`.
6. In the Resources page, select **Advanced Requirements** > **Resource Groups** > **Add** to associate the job definition to the resource group created previously.
7. Write the resource group name.
8. Click **Save** to store the job definition.

   **Note:** You can also define optimization policies for your resource group. For further information, see *IBM Workload Scheduler: Scheduling Workload Dynamically*.
9. Select the new JSDL and upload it to the server by clicking on the corresponding icon.
10. Submit the job in one of the following ways, depending on whether you want to submit it as a broker job or by using a Tivoli Workload Scheduler job definition.
    - Log in to the Dynamic Workload Console and choose the **Tivoli Dynamic Workload Broker** portfolio option.
      – Select **Definitions** > **Jobs** Optionally specify search criteria and click **Search**. Select the job definition you created in the previous steps.
      – To run the job, select **Submit** and click **Go**.
    - Log in to the Dynamic Workload Console and choose the **Tivoli Workload Scheduler** portfolio option.
      – Select **Workload** > **Design** > **Create Workload Definitions**. In the Workload Designer create a new broker job definition filling in the

required fields as appropriate. In the **Workload Broker Job Name** field type the name of the JSDL file you created in the previous steps.

– Submit the IBM Workload Scheduler job by selecting **Workload** > **Submit** > **Submit Predefined Jobs** .

You can also add the job to an existing job stream , or submit the job using the **jobsubmit** command. For further information about the dynamic workload broker command-line interface, see *IBM Workload Scheduler: Scheduling Workload Dynamically*.

## Expected Results

When the job is submitted, the broker application selects an agent from those defined in the resource group, based on availability and other criteria which you can specify.

When multiple instances of the job are submitted using the selected JSDL, IBM Workload Scheduler sends them dynamically to the agents defined in the resource group. Regardless of the agent selected, the job is run on the same SAP R/3 system.

## Sample Configuration files

An example jsdl file for the job:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdlxa="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlxa" name="sapjobs">
 <jsdl:variables>
   <jsdl:stringVariable name="optionFile">SAP_SYS</jsdl:stringVariable>
 <jsdl:variables>
<jsdl:application name="xajob">
 <jsdlxa:xajob accessMethod="r3batch.exe" error="C:\myerr.txt"
     output="C:\myout.txt" target="${optionFile}">
      <jsdlxa:taskString>-job DYN_SUB -C A -flag type=exec -user TWS4APPS1
        -s1 type=A -s1 program=Z_TWS_SLEEP -v1 ONE </jsdlxa:taskString>
      <jsdlxa:credential>
        <jsdlxa:userName>fta85bro</jsdlxa:userName>
        <jsdlxa:password>fta85bro</jsdlxa:password>
      </jsdlxa:credential>
    </jsdlxa:xajob>
  </jsdl:application>
  <jsdl:resources>
    <jsdl:group name="SAP_ENV"/>
  </jsdl:resources>
</jsdl:jobDefinition>
```

An example options file:

```
R3CLIENT=000
R3HOST=9.168.125.153
R3INSTANCE=00
R3PASSWORD=&p2N-l?bO-dF3F-uX
R3SID=NSP
R3USER=twsuser
LONG_INTERVAL=300
R3AUDITLEVEL=0
SHORT_INTERVAL=10
retry=-1
xbpversion=3
pchain_details=on
rfc_open_retry=-1
ccms_alert_history=ON
```

```
throttling_enable_job_interception=ON
throttling_send_ccms_data=ON
throttling_interval=30
throttling_send_ccms_rate=2
```

# Defining and scheduling new and classic jobs with dynamic capabilities

This scenario describes how you can use the new workstations with dynamic capabilities to run the jobs you created for the previous IBM Workload Scheduler workstations. To run these jobs on the new workstations, you must change only the workstation where you want the job to run. You can also define new job types. While the classic IBM Workload Scheduler job is a generic script or command, you can now define specific job types, such as database or Java™ jobs. To create this type of jobs, you do not need specific skills on the applications where the job runs.

## Business Goal

An insurance company runs a number of jobs at night to save the data processed during the day to the backup database. It must also gather all of the data related to the transactions completed during the day from all the workstations in the company branches. They use DB2® databases. Using the Workload Designer, they create a job to perform a database backup and another job to extract the data for the daily transactions. To perform these operations, they use the new database job type.

After gathering data from all the company workstations, they copy the resulting data on a single workstation and process it to generate a report. They choose the best available workstation dynamically by defining the requirements necessary to run the job: a workstation with Windows operating system installed, low processing workload, and the program required to generate the report.

The report highlights how many new contracts were signed and how many customers are late with their payments. A mail is sent to the chief accountant, listing the number of new contracts and late customers. A mail is also sent to the secretaries with the name of the late customers for each branch, so they can write them a letter pointing out the delay in the payment.

If the administrator does not want to modify the job stream he used in the past to run a Java job, he can, for example, modify the name of the workstation where he wants the job to run, inserting the name of a pool or dynamic pool of dynamic agents where the Java executable is installed. IBM Workload Scheduler translates the syntax of the job so that it can be run by the Java executable and assigns the job to the best available resource in the pool.

The company can reach this objective, by:

- Using the new workstations with dynamic capabilities to run the jobs the administrator created for the previous IBM Workload Scheduler workstations, the administrator changes only the workstation where he wants the job to run. The major advantage is that he can use the workflows he previously created without additional effort.
- Defining the job types listed without having specific skills on the applications where the job runs:
  - Database jobs
  - File transfer jobs

- Web services jobs
- Java jobs

These job types run on the new workstations:

**dynamic agents**
Workstations capable of running both classic and new job types.

**pools** Groups to which you can add dynamic agents depending on your needs. Jobs are assigned dynamically to the best available dynamic agents.

**dynamic pools**
Groups of dynamic agents for which you specify your requirements and have IBM Workload Scheduler select the dynamic agents which meet your needs. Jobs are assigned dynamically to the best available dynamic agent.

### Roles

This section lists the user roles needed to run the scenario:

**IBM Workload Scheduler Administrator**
Manages the workload by creating and modifying jobs.

**IBM Workload Scheduler Job Scheduler**
Manages the workload by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- A IBM Workload Scheduler network with the Dynamic Scheduling capability.
- A Dynamic Workload Console

### Setting up the environment

Complete the tasks described in "Setting up your environment for dynamic scheduling" on page 3.

## Running the Scenario

To complete this scenario, you need to perform the following operations:

### About this task

- Create a database job
- Create a dynamic pool with the following requirements:
  - Windows operating system installed
  - Program required to generate the report
  - Low processing workload
- Modify an existing job stream so that it can run on an dynamic agent.

To create a database job, perform the following steps:

### Procedure

Log on to the Dynamic Workload Console and launch the **Workload Designer**.

- To create a database job, perform the following steps:
  1. In the **Working List** pane, select **New ↦Job Definition ↦**

2. Select **Database**.
3. Fill in the fields as necessary.

- To modify an existing job stream so that it can run on an dynamic agent, perform the following steps:
  1. Click **Search** ↦ **Job Stream** and type the name of the job stream you want to modify.
  2. Select the job stream and click **Edit**. The job stream is displayed in the right pane.
  3. Click the Browse icon next to the **Workstation** field. The **Lookup** panel is displayed.
  4. Search for the workstation where you want the job to run, select it and click **OK**.
  5. Save the job stream. A new job stream is created in the database.
  6. Optionally delete the previous job stream.

## Creating a dynamic pool

### About this task

To create a create a dynamic pool with the necessary requirements, perform the following steps:

### Procedure

1. Log on to the Dynamic Workload Console.
2. From the navigation toolbar, select the entry for **Administration** > **Workload Environment Design** > **Create Workstations**.
3. Select an engine and click **Create Workstation**. The **Workstation properties** panel is displayed.
4. In the **Workstation type** menu, click the **Dynamic Pool** item. The layout of the panel changes slightly.
5. Fill in the fields as necessary.
6. Click **Edit Requirements**. The **Requirements** panel is displayed.
7. Select the **Windows** check box.
8. Add a logical resource you previously created where the required product is installed.
9. Select the **Workload Balance** radio button.

## Expected Results

You have created a job which performs operations on a database, a second job that is dynamically assigned to the best available dynamic agent and also modified an existing job stream with minimal effort to take advantage of dynamic scheduling functions.

# Chapter 3. Cloud environment scenarios

The following scenarios show how to use IBM Workload Scheduler with cloud environments to take advantage of virtualization, flexibility, and standardization.

## Managing workload in dynamic environments

This scenario shows how Workload Automation in modern IT infrastructures supports dynamic environments, where new servers can be provisioned and removed in just a few minutes by using Cloud and Virtualization. Providing new applications to support business requires a high level of automation for the verification of new releases in a production-like environment. The change management of the workflows can be simple and intuitive, making it possible to share pre-built assets across departments and organizations.

### Scenario goal

This scenario shows how an integrated Development and Operations Lifecycle is an agile, scalable, and flexible solution for end-to-end lifecycle management and automation, creating an environment that has collaboration between development and operation teams reducing the possibility of applications being returned due to deployment issues and achieving optimal efficiency. This solution is important to the business because every delay in getting a software system deployed carries a lost opportunity cost or a financial risk. Any production outage of any duration can have a direct, negative impact on a company's revenue. Outages related to software defects carry additional risk due to the coordination of resources required to identify, analyze, and correct the problem, as well as to test the fix and deploy it into production without breaking dependent applications or components.

#### Business Scenario

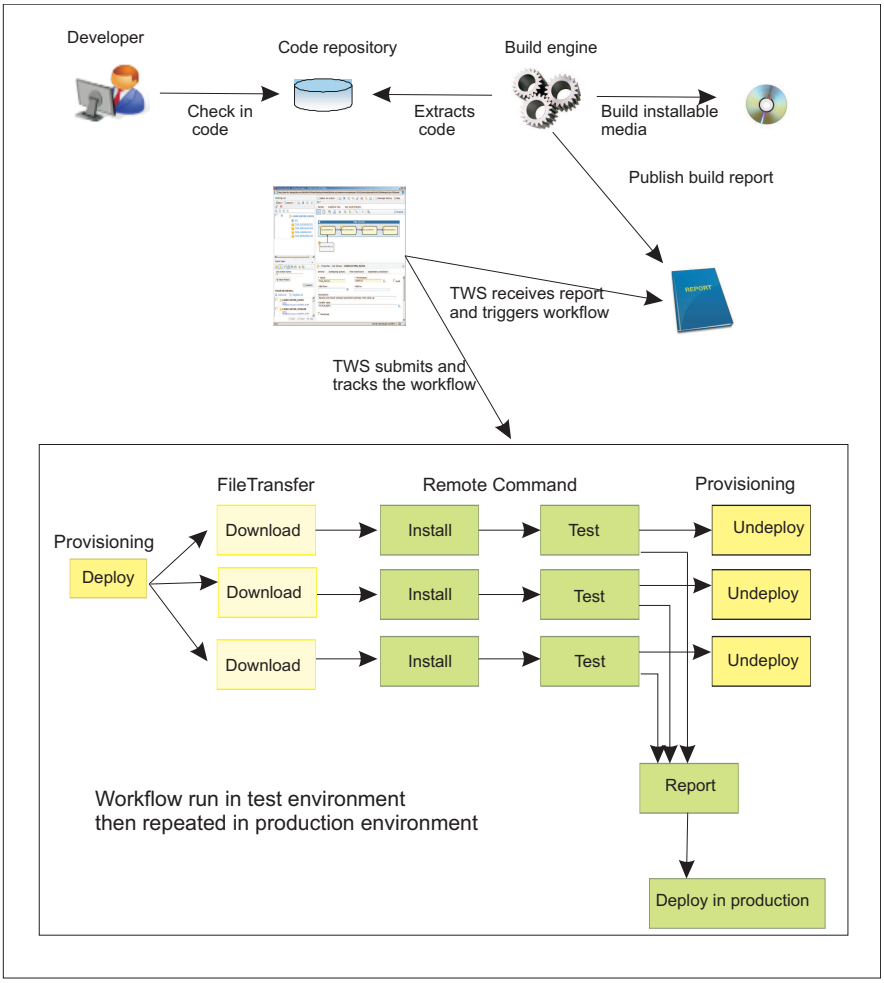
A company producing and delivering software products finds it difficult to realize its revenue because deployment fails due to inaccuracies in the configuration or is delayed due to manual processes. The business cannot benefit from the new capabilities and suffers financially if the failure or delay impacts business continuity or compliance. Because the organization has complex manual deployment processes and, as a result, high deployment failure rates, it needs to find a solution. It currently manages configuration specifications in spreadsheets and has difficulty in keeping the information current. The final part of the deployment of complex applications can often take several months.

By deploying an integrated solution that uses IBM Workload Scheduler with Provisioning and Remote Command job types, the company can reduce the cycle time of a release to a number of days or even hours.

Using IBM Workload Scheduler, the company defines a Provisioning job to connect to a SmartCloud Provisioning server and deploy all the virtual machines it needs for any specific software project. Then, defining File Transfer and remote command jobs, it can download, install and test the software on different operating systems without installing specific agents to run the commands.

Finally, after the software is tested, the unnecessary virtual machines can be easily removed by running Provisioning jobs.

After running this workflow in a test environment, the company can easily repeat the same flow in a production environment.




## Roles

This section lists the user roles needed to run the scenario:

**IBM Workload Scheduler Developer**
Defines the jobs.

**IBM Workload Scheduler Job Scheduler**
Manages IBM Workload Scheduler workload by submitting and monitoring jobs.

# Software Requirements

The following software must be installed and configured, before running this scenario:

- IBM Workload Scheduler version 8.6.0.2 network with the Dynamic Scheduling capability
- IBM SmartCloud Provisioning 2.1

## Before you begin

Ensure that you have performed the prerequisite steps described in Prereq_scp_job_t.html

## Running the Scenario

To complete the scenario, perform the following steps:

### About this task

### Procedure

1. Create the virtual images of the SmartCloud Provisioning environment.
2. Create a Provisioning job to deploy several instances of the virtual images defined in previous step.
   a. In the Working List pane, select **New** > **Job Definition** > **Cloud** > **Provisioning**.
   b. In the Properties pane, specify the properties for the job definition you are creating using the tabs available. For more detailed information about the UI elements on each tab, see the Dynamic Workload Console online help.
   c. On the **General** tab, specify general information about the job definition.
   d. On the **Affinity** tab, optionally, specify the affinity relationship between two or more jobs. Affinity relationships cause jobs to run on the same workstation as the affine job.
   e. On the **Recovery Options** tab, optionally specify the recovery options to be followed if the job abends. You can choose to stop or continue the scheduling activity, rerun the job, to display a prompt or run a recovery job.
   f. On the **Connection Server** tab, specify the names and passwords of the authorized users associated with the IBM SmartCloud Provisioning server.
   g. On the **Actions** tab, select **Deploy** to deploy a virtual image in the cloud group and create a new virtual system instance containing the number of virtual image instances you specify.
3. Create a Provisioning job to start the systems you have deployed.
4. Create a **FileTransfer** job to upload the new build and all the required software on the deployed systems.
   a. From the Dynamic Workload Console portfolio, click **Workload** > **Design** > **Create Workload Definitions**
   b. Specify an engine name, either distributed or z/OS. The Workload Designer opens. Job types and characteristics vary depending on whether you select a distributed or a z/OS engine.
   c. In the Working List pane, select **New** > **Job Definition** > **File Transfer and Coordination** > **FileTransfer**.
   d. In the Properties pane, specify the properties for the job definition you are creating using the tabs available. For more detailed information about the UI elements on each tab, see the Dynamic Workload Console online help.
5. Create a Remote Command job to complete the deployment of the application and run a set of tests
   a. In the Working List pane, select **New** > **Job Definition** > **Native** > **Remote Command**.
   b. In the Properties pane, specify the properties for the job definition you are creating using the tabs available. For more detailed information about the

UI elements on each tab, see the Dynamic Workload Console online help. The first tabs are common to Provisioning.

c. On the **Tasks** tab, enter the software installation and test commands that you want to run.

d. On the **Environment** tab, optionally specify the standard output, and standard error files for the command. These files are stored on the agent, not locally on the workstations where the command runs. ensure you have write permission for the specified directories, otherwise no file will be created.

6. Create a Provisioning job to delete the unnecessary virtual instances

   a. In the Working List pane, select **New** > **Job Definition** > **Cloud** > **Provisioning** and create a new Provisioning job specifying in the **Actions** tab, to delete the unnecessary virtual instances.

7. Create a job stream with the necessary job dependencies to run the jobs in the appropriate sequence.

## Expected result

You have created a Provisioning job that deploys some virtual instances, and a Remote Command job that installs and verifies the built software and generates a success report. Finally, when the environment is no longer needed, you can run a Provisioning job to remove the virtual instances that you no longer use.

# Sharing and reusing standard workload process templates

This scenario shows how, after developing a workload process that efficiently answers your needs, you can standardize it by virtualizing the workload, regardless of its topology, and exporting it to a new environment where it can be easily deployed.

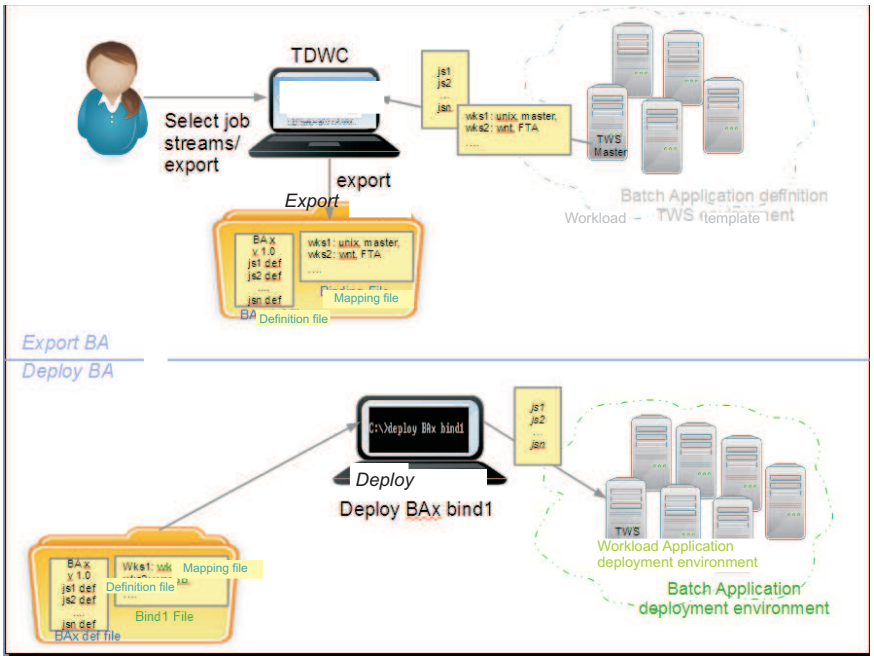
## Scenario goal

Successful, efficient workload processes can be reused in multiple environments. For example, you might want to move your tested and fine-tuned workload from your development to your production environment. Also, you might have standardized a solution that can work in your local environment as well as in all your branch offices. Alternatively, you might want to produce a set of standard, virtualized solutions that can be commercialized or shared with a community of IBM Workload Scheduler users.

### Business Scenario

A service provider that produces and delivers software products for the banking sector has defined a workload application to detect fraud in banking financial flows. After defining a set of job streams containing the jobs and dependencies to detect and block fraud attempts, the service provider includes these job streams in a workload application and exports it as a compressed file. The service provider then makes this workload application available to the banking sector. An important central bank that has been struggling against fraud problems for years, purchases the workload application from the service provider. To run the workload application, the bank just has to customize a mapping file to specify the names and information typical of its environment and then deploy the workload application. The same fraud-detecting workload, fine-tuned by the service provider, is now running in the bank environment. The central bank can share this workload application with all its branch offices to ensure all banking operations are safeguarded against financial frauds.

The same workload application can be stored on a server and shared with the whole authenticated banking community to be further improved by the contribution of other users who adapt it to the latest international tax regulations. As a result, the workload application might also become a type of open-source IBM Workload Scheduler product that is continuously improved and run by its users.




## Roles

This section lists the user roles required to run the scenario:

**IBM Workload Scheduler Developer**
Defines the jobs.

**IBM Workload Scheduler Job Scheduler**
Manages IBM Workload Scheduler workload by submitting and monitoring jobs.

# Running the Scenario

To complete the scenario, perform the following steps:

## About this task

## Procedure

1. Create the jobs, job streams, and dependencies that satisfy your requirements.
2. Create and export the workload application template by performing the following steps:
   a. In the Working List pane, select **New** > **workload application template**. The workload application template is created in the Details view and its properties page is displayed.
   b. In the Properties pane, specify the properties for the workload application template that you are creating, using the available tabs. For more detailed information about the UI elements on each tab, see the Dynamic Workload Console online help.

c. Click **Save** to save the Dynamic Workload Console in the database.
d. From the Details view, right-click the Dynamic Workload Console and add the created job streams to it. Together with the job streams, the corresponding dependencies are also automatically added to the Dynamic Workload Console.
e. Right-click the Dynamic Workload Console and click **Export** to produce a compressed file containing all the files and information required to make the workload to run also in another environment.

## Results

The compressed file contains:

*workload application template name*_**Definitions.UTF8.xml**
XML file that contains the definitions of all the exported objects. These definitions will be deployed in the target environment so as to populate the target database with the same objects existing in the source environment.

*workload application template name*_**Mapping.UTF8.properties**
Mapping file that the target user will modify replacing the names of the objects in the source environment with the names that these objects will have in the target environment.

*workload application template name*_**SourceEnv_reference.txt**
Reference information containing the definitions of the workstations used in the workload application and other information that can be useful to correctly map the source environment into the target environment and allow the workload application to run.

## Deploying the workload application

### About this task

The central bank must then deploy the workload application template in its environment, creating all the required objects to run the workload. Deploying a workload application template is a two-step process beginning with customizing the mapping file by specifying the names of the objects as they are defined in the target environment (for example, the names of the workstations on which the job streams will run) and then importing the mapping file and definitions file into the new IBM Workload Scheduler environment. All the import and customization steps must be performed in the target environment by using the command line.

### Procedure

1. Customize the mapping file. Assign to each object listed in the mapping file the name it must have in the target environment. For example:

```
CALENDAR_GALAXY=WORKINGDAYS
JOBSTREAM_JSA01=BANK1
JOBSTREAM_JSA02=BANK2
JOBSTREAM_JSA03=PAYROLL
JOBSTREAM_JSA04=ACCOUNTS
JOBSTREAM_JSA05=FOREIGN
JOB_JOB1=PAY
JOB_JOB2=CASH
JOB_JOBDEF6=HOMEOFFICE
RESOURCE_RES1=ROME1
VARIABLEVALUE_VALE_PARAM1=date
VARIABLEVALUE_VALE_PARAM2=root
VARTABLE_VALE=VALE
WORKSTATION_WSNAME=UNIX
```

The mapping file creates a relationship between objects in the source environment, where the workload application is created, with objects in the target environment. If you specify a target object name that is the same as an object already in the target environment, then the import operation fails.

2. From the command line, submit the following command, indicating the file names of both the definitions file and the customized mapping file:

   ```
   -import  <wkldappname_Definitions.UTF8.xml> [wkldappname_Mapping.UTF8.properties]
   ```

   where,

   ***wkldappname***
   Represents the name of the workload application you want to import.

   For more information about the command usage and syntax, see wappman in the *User's Guide and Reference*.

## What to do next

You can subsequently update a workload application if a newer version becomes available. Any objects already present in the IBM Workload Scheduler database of the target environment are replaced with the updated versions, any objects that do not already exist in the target environment are created, and objects are deleted from the target environment if the object definition has been removed from the updated workload application. The same mapping file used to originally deploy the workload application can be used to update it, customizing any new objects being deployed with the update.

If you need to delete an object from the workload application in the target environment, you must import an updated version of the workload application removing the definition of the object you want to delete from it so that the object in the source environment is deleted also.

In the future, the central bank might decide to share the workload application with a community of IBM Workload Scheduler users belonging to the banking sector to keep it as an alive and continuously-improving application.

# Chapter 4. Workload scheduling rules scenario

The following scenario shows how run cycle groups add additional flexibility to define when your workload is scheduled to run.

A run cycle group is a distinct database object and the same run cycle group can be reused in different job streams, avoiding the necessity of having multiple run cycles definitions to have the same scheduling rules.

Run cycle groups are especially useful when you need to identify the days when a job stream would normally be scheduled to run, but for some reason it is not required to run so it can be "excluded". Run cycles are organized in subsets within the run cycle groups so that the exclusive run cycles are applied against the positive occurrences generated by the run cycles belonging to the same subset.

In addition to specifying inclusive and exclusive run cycles, you can also specify a positive or negative offset on a run cycle group.

## Inclusive and exclusive rule-based run cycles

This scenario shows how a IBM Workload Scheduler administrator can use subsets of run cycles in a run cycle group to achieve a specific workload schedule that, without the use of a run cycle group, would require the definition of numerous run cycles, calendars, or both. In addition, it demonstrates how you can reuse run cycle groups in different job streams and add an offset to shift a workload schedule forward by one day.

### Scenario goal

In this scenario, a IBM Workload Scheduler administrator needs to run a process that elaborates sales projections every day except Friday, and on the first day of each month; in the latter case, it must run even if the first day of the month is a Friday. The administrator must then run a post-sales check according to the same schedule but a day later.

#### Business Scenario

To define the required workload schedule, the IBM Workload Scheduler administrator creates a run cycle group with two subsets. The first subset defines that the process must run every day except Fridays, and the second subset defines that the process must run on the first day of each month, even if the first day of the month is a Friday. The same run cycle group is reused in a different job stream to run a post-sales check, however, this job stream should follow the same schedule as the sales projections, but one day later.

#### Roles

This section lists the user roles needed to run the scenario:

**IBM Workload Scheduler Developer**
Defines the run cycle groups in a job stream.

**IBM Workload Scheduler Job Scheduler**
Manages IBM Workload Scheduler workload by submitting and monitoring jobs.

## Software Requirements

The following software must be installed and configured, before running this scenario:

- IBM Workload Scheduler version 9.1.
- Dynamic Workload Console version 9.1.

## Setting up the environment

To run this scenario you must install or upgrade to IBM Workload Scheduler version 9.1.

## Running the Scenario

To complete the scenario, perform the following steps:

### About this task

### Procedure

1. Define the run cycle group that will contain the two subsets.
   a. In the Working List pane, select **New** > **Run cycle group**.
   b. In the Properties pane, specify the properties for the run cycle group definition you are creating using the tabs available. For more detailed information about the UI elements on each tab, see the Dynamic Workload Console online help.
   c. On the **General** tab, specify general information about the run cycle group definition such as the name, `SalesProjections`.
   d. On the **Time restrictions** tab, specify time restrictions for the job stream associated to the run cycle group.
   e. Save the run cycle group definition.
2. Create a subset within which to create two run cycles. The first to run the process on a daily basis, and the second to exclude Fridays.
   a. Right-click the run cycle group `SalesProjections` and click **Add Subset**. By default, the subset name is `SUBSET-1`, but you can optionally modify the name.
   b. Right-click `SUBSET-1` and click **Add Run Cycle**.
   c. On the **General** tab, specify general information about the run cycle definition such as the name, `DailyProcess`.
   d. Leave the default selection for **Inclusive or Exclusive Run Cycle** as Inclusive.
   e. On the **Rule** tab, specify **Daily** for the **Repeat schedule**, and leave the default selection **1** for **Run every selected number of days** and **Everyday** for **On the following day type**.
   f. Save the run cycle group definition.
   g. Right-click `SUBSET-1` again and click **Add Run Cycle** to add a second run cycle that excludes Friday from the daily process.
   h. On the **General** tab, specify general information about the run cycle definition such as the name, `NoFridays`.
   i. In **Inclusive or Exclusive Run Cycle** select **Exclusive** since this run cycle serves to exclude Fridays.
   j. On the **Rule** tab, specify **Weekly** for the **Repeat schedule**, and select **Friday** for **On the following days of the week**.

k. Save the run cycle group definition. You can click the **Run Cycle Preview** tab to view the schedule for each of the run cycles and then the combination of the run cycles on a calendar.

3. Create a second subset to define a run cycle to run the process on the first day of each month.
   a. Right-click the run cycle group `SalesProjections` and click **Add Subset** to add a second subset. By default, the subset name is `SUBSET-2`, but you can optionally modify the name.
   b. Right-click `SUBSET-2` and click **Add Run Cycle**.
   c. On the **General** tab, specify general information about the run cycle definition such as the name, `FirstDayoftheMonth`.
   d. Leave the default selection for **Inclusive or Exclusive Run Cycle** as Inclusive.
   e. On the **Rule** tab, specify **Monthly by month day** for the **Repeat schedule**, and **1st Day** for **On the (ascending order)**.
   f. Save the run cycle group definition.
4. Create a job stream containing the jobs to run the sales projections processes and then associate the run cycle group, `SalesProjections`, by adding a run cycle containing a repeat schedule set to **Run Cycle Group**.
   a. From the **Working List** click **New** > **Job Stream**.
   b. Define a name and the workstation for the job stream.
   c. Click **Select an Action** > **Add Jobs**.
   d. Search for and select the jobs related to the sales projections processes and click **Add**.
   e. Select the job stream in the **Details** view, and click **Select an Action** > **Add Run Cycle**.
   f. Define a name for the run cycle.
   g. On the **Rule** page, select **Run Cycle Group** for the **Repeat schedule**.
   h. Search for and select the run cycle group `SalesProjections` and click **OK**.
   i. Save the job stream definition Click **Run Cycle Preview** to view a calendar displaying the run cycle schedule for the job stream.
5. Reuse the same run cycle group in a different job stream that runs a post-sales check according to the same run cycle schedule but a day later.
   a. Define a new job stream or open an existing job stream containing the jobs to run the post-sales check, and add a run cycle to the job stream specifying the `SalesProjections` run cycle group as the rule for the job stream.
   b. In the **Offset** section on the **Rule** page, click the up arrow to specify an offset of 1 day. Click **Run Cycle Preview** to display the same run cycle schedule used in the sales projections job stream but with the dates shifted over by one day.

## Expected result

You have created a job stream with a run cycle group containing two subsets that create a workload schedule that runs a process every day except Friday and the first day of each month, even if the first day falls on a Friday. The same run cycle group is used to run a different process but a day later with respect to the first job stream.

# Chapter 5. Maintaining dependencies across different scheduling environments

You can create relationships between jobs even when the jobs run in different scheduling environments.

A cross dependency is a dependency of a local job on a remote job running in a different scheduling environment. It is achieved by using a shadow job, which runs in the same environment as the local job and maps the remote job processing.

The scenarios in this section demonstrate how you can migrate a workload to run in two different scheduling environments, maintaining any dependencies created between jobs, as well as a scenario that demonstrates how you can synchronize the workload of two scheduling environments.

## Migrating a workload to another IBM Workload Scheduler scheduling environment while maintaining dependencies

This scenario describes how you can split the workload between different IBM Workload Scheduler distributed environments, keeping the dependency flow. This situation can happen, for example, when specific activities are migrated to a newly added IBM Workload Scheduler master domain manager.

### Business Goal

Two insurance companies are merged. Each of them has implemented its own IBM Workload Scheduler distributed solution to run a workload. As result of the merge, to avoid workload duplication and unify processes, some activities must be migrated from one master domain manager to the other.

To accomplish this task, the administrator of TWSA needs to ensure that the workflow dependencies are maintained even though part of the activities are moved to TWSB managed environment.

Figure 1 on page 36 shows the workload flow that must be migrated from TWSA to TWSB:

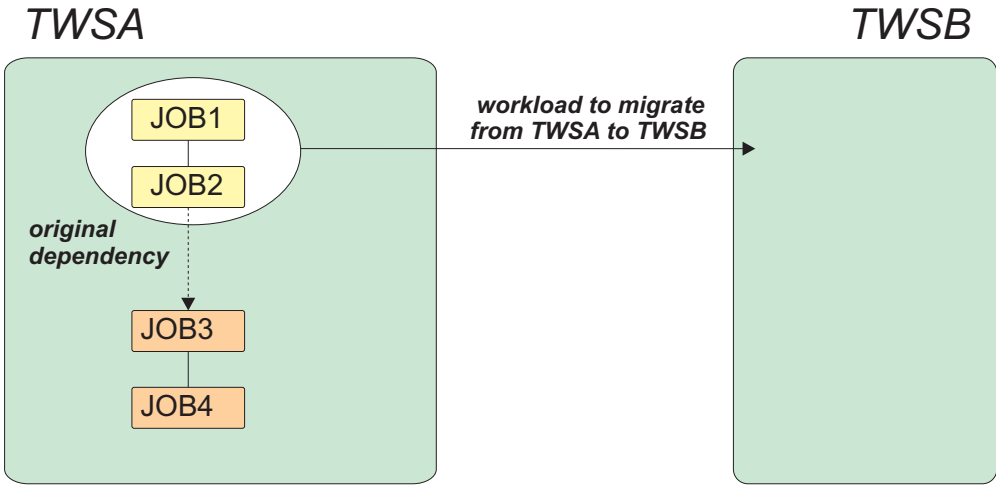



*Figure 1. Processing flow to migrate*

## Roles

This section lists the user roles needed to run the scenario:

**IBM Workload Scheduler Administrator**
Manages IBM Workload Scheduler workload by creating and modifying jobs.

**IBM Workload Scheduler Operator**
Manages IBM Workload Scheduler workload by submitting and monitoring jobs.

## Software requirements

The following software must be installed and configured, before running this scenario:

- Two IBM Workload Scheduler networks.
- A Dynamic Workload Console

## Setting up the environment

To run this scenario do the following:

**On `TWSA`**

- Ensure that the version of IBM Workload Scheduler installed is at least 8.6.
- Define remote engine workstations that map `TWSB` master domain manager, and possible backup masters, and a dynamic pool workstation that contains all these remote engine workstations. This configuration helps you in managing, in a transparent way, failover situations or maintenance activities of `TWSB` master domain manager.
- Verify that you can successfully `telnet` to the `TWSB` system using the port number specified in the remote engine workstations definition.

**On `TWSB`**

- Ensure that the version of IBM Workload Scheduler installed is at least 8.6.

- If you specified in the global option, bindUser, a user different from the *TWS_user*, ensure that the specified user has the required authorizations to bind the job stream containing JOB2, as well as JOB2 itself, in the plan.
- Ensure that the resources that were needed to run JOB1 and JOB2 on TWSA are available with the same naming convention on TWSB.

# Running the Scenario

## About this task

To complete this scenario, as TWSA Administrator, you need to perform the following operations:

1. Migrate JOB1 and JOB2 from TWSA to TWSB.
2. Remove the dependency on JOB2 from the JOB3 definition.
3. Define a shadow job on TWSA , JOB2SH, pointing to JOB2 running on TWSB
4. Define a dependency on JOB2SH for JOB3 on TWSA.

Figure 2 shows the new workload flow between the two environments:

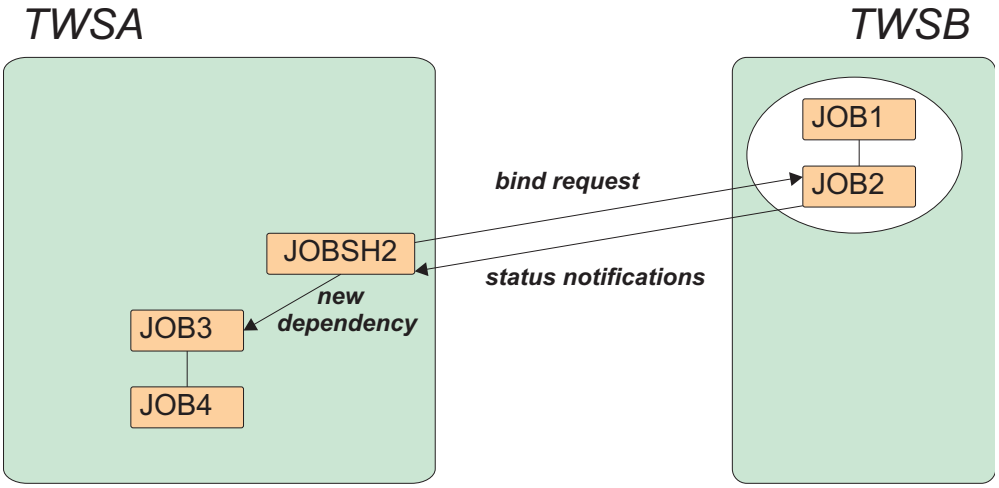



*Figure 2. New processing flow*

As TWSB Administrator, generate the plan on TWSB.

As TWSA Administrator, generate the plan on TWSA.

Then, as TWSA Operator, do the following:

1. Log in to the Dynamic Workload Console to monitor the workload flow.

   **Note:** If the engine connection to TWSB was defined on the Dynamic Workload Console, then the TWSA Operator can monitor both the status of JOB2 defined on TWSB, and the status of the shadow job JOB2SH defined on TWSA, from the same user interface.
2. Monitor when the status of the shadow job becomes **BOUND**. This means that the association of JOB2SH with the instance of JOB2 was established.
3. Rerun and recovery of the JOB2 instance are managed automatically by the product and mapped to the shadow job status. If JOB2SH is submitted successfully, but then goes to **ERROR**, then you have to contact the TWSB administrator to understand why the bind failed.

### Expected Results

You moved and automated the workflow dedicated to run an activity from one IBM Workload Scheduler environment to another, maintaining the workflow dependency.

# Creating dependencies in a workload running in two IBM Workload Scheduler scheduling environments

This scenario describes how you can create a dependency relationship in a workload running in different IBM Workload Scheduler environments using cross dependencies.

## Business Goal

An insurance company runs a number of jobs to collect the data processed during the day. The information about the transactions completed during the day by all of the workstations in the company branches is collected and processed on the distributed IBM Workload Scheduler master domain manager, TWSD.

As soon as this processing completes, the IBM Workload Scheduler for z/OS controller at the head office, TWSZ, must run reports and statistics on this data to compare the actual results with the projected results.

To accomplish this task, the IBM Workload Scheduler administrator of TWSZ needs to automate the following process: the job RUNREP that runs the reports starts only after the job COLLECT_TRANSACTIONS, that collects data processed at the branch offices, completes successfully on TWSD.

### Roles

This section lists the user roles required to run the scenario:

**IBM Workload Scheduler Administrator**
Manages IBM Workload Scheduler workloads by creating and modifying jobs.

**IBM Workload Scheduler Operator**
Manages IBM Workload Scheduler workloads by submitting and monitoring jobs.

### Software requirements

The following software must be installed and configured, before running this scenario:

- A IBM Workload Scheduler for z/OS network and a IBM Workload Scheduler network, whose managers can ping each other.
- A Dynamic Workload Console.

### Setting up the environment

To run this scenario do the following:

**On TWSZ**

- Ensure that the version of IBM Workload Scheduler installed is at least 8.6.

- Define in the **ROUTOPTS** initialization statement, an HTTP or HTTPS destination for the TWSD master domain manager and for each backup master defined.
- Define remote engine workstations for each of these destinations. The primary remote engine workstation, TWSDREM points to the TWSD master domain manager. The other remote engine workstations are defined cascading as alternate workstations.

**On TWSD**

- Ensure that the version of IBM Workload Scheduler installed is at least 8.6.
- If you specified in the global option bindUser a user different from *TWS_user*, ensure that the specified user has the required authorizations to bind the job stream containing COLLECT_TRANSACTIONS as well as the COLLECT_TRANSACTIONS job itself.

# Running the Scenario

## About this task

To complete this scenario, as the TWSZ Administrator, you need to perform the following operations:

1. Double-check with the TWSD Administrator the information needed to identify the specific COLLECT_TRANSACTIONS job instance in the TWSD plan.
2. Create a shadow job COLLECTSH with the following characteristics:
   - It is defined on the TWSDREM remote engine workstation.
   - It points to the COLLECT_TRANSACTIONS job instance.
   - The input arrival of the shadow job must follow the input arrival of the COLLECT_TRANSACTIONS job instance.
3. Define a dependency in RUNREP job from COLLECTSH.
4. Extend the plan.

Figure 3 shows the workload flow between the two environments:

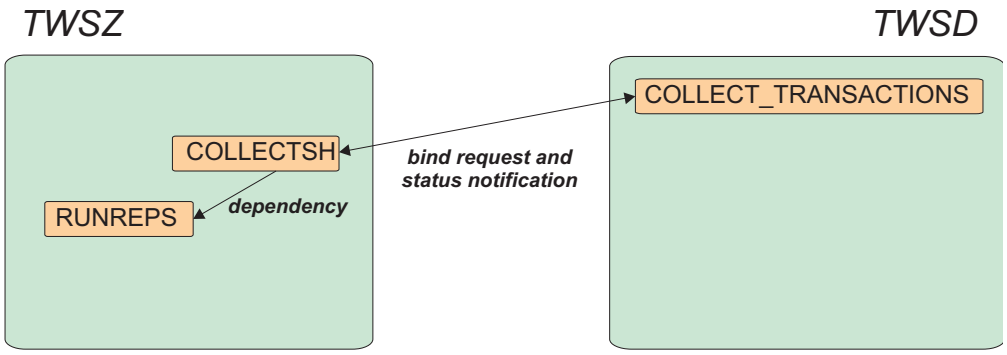



*Figure 3. Processing flow*

Then, as TWSZ Operator, do the following:

1. Log in to the Dynamic Workload Console to monitor the status of the shadow job COLLECTSH.

   **Note:** If the engine connection to TWSD is defined on the Dynamic Workload Console, then the TWSZ Operator can monitor both the status of

COLLECT_TRANSACTIONS job defined on TWSD and the status of the shadow job COLLECTSH defined on TWSZ from the same user interface.

2. Monitor when the status of the shadow job becomes **BOUND**. This means that the association of COLLECTSH with the instance of COLLECT_TRANSACTIONS was established.
3. Rerun and recovery of the COLLECT_TRANSACTIONS instance are managed automatically by the product and mapped to the shadow job status. If COLLECTSH is submitted successfully, but then fails or goes to **ERROR**, you have to contact the TWSD administrator to verify what went wrong in the COLLECT_TRANSACTIONS instance.

As soon as the information collected at the branch offices is collected, the COLLECT_TRANSACTIONS job completes successfully on TWSD. As a consequence, the shadow job COLLECTSH also completes on TWSZ. The RUNREP dependency from COLLECTSH is resolved and, if RUNREP is free from other dependencies, it can start running the report on closing day activities.

## Expected Results

You automated your closing day activities run overnight by creating a dependency on a job running locally from a job, that collects data from the branch offices, running on a different IBM Workload Scheduler environment.

# Chapter 6. Variable Tables

IBM Workload Scheduler Variable Tables describe how to customize job and job stream behavior.

## Customizing jobs and job streams based on when they are scheduled to run

This scenario shows how to customize jobs and job streams based on when they are scheduled to run, that is on which days they run, using variable tables referenced at the run cycle level.

By having the ability to manage different tables of variables in run cycles, you no longer need to define two different job streams to run the same work on different data. Instead, you define:

- A single job stream with two run cycles, referencing two different variable tables.
- File dependencies on jobs using variables, forcing instances generated on a specific day to use different variable values with respect to instances generated on another day.
- A variable in the path of the **scriptname** job attribute.

Using variable tables you reduce the number of scheduling object definitions required to implement your workload saving time and money.

Using variable tables you need to define less scheduling objects required to implement your workload with respect to the past saving time and money.

You can find more information on this subject in the *User's Guide and Reference*.

### Scenario goal

A sales department needs to collect sales data to produce reports both at a business unit level and at a corporate level. The department manager needs reports both weekly and monthly. Report data is stored in two different directories. Data for weekly reports is stored in a file located in the `/reports/weekly` directory. Data for monthly reports is stored in a file located in the `/reports/monthly` directory.

The files containing the input data needed to generate the reports are stored in two different directories, and the job stream used to generate reports has a dependency on these files. To collect data you can now create one job stream with two different run cycles (one that runs weekly and one that runs monthly) that reference two different variable tables containing a variable that is in the file path.

#### Roles and skills

This section lists the users involved in the scenario, the related roles, and the required skill level:

**IBM Workload Scheduler Job Scheduler**
: Manages Tivoli Workload Scheduler workload. Required skills include Tivoli Workload Scheduler knowledge.

### System requirements

Install the following software before starting the scenario:

- IBM Workload Scheduler version 8.5
- Dynamic Workload Console version 8.5

### Setting up the environment

Complete the following tasks before running the scenario:

- Install and configure IBM Workload Scheduler version 8.5
- Install and configure Dynamic Workload Console version 8.5

## Running the Scenario

To complete the scenario, perform the following steps:

### Procedure

1. Define the SC1_WEEKLY_DATA_TABLE and the SC1_MONTHLY_DATA_TABLE variable tables as follows:

```
VARTABLE SC1_WEEKLY_DATA_TABLE
  MEMBERS
  REP_PATH "/reports/weekly"
END

VARTABLE SC1_MONTHLY_DATA_TABLE
  MEMBERS
  REP_PATH "/reports/monthly"
END
```

2. Locate the jobs used to generate the reports. The jobs run a script which receives the directory name as an input argument as shown in the syntax below:

```
SC1_PARSE_DATA SCRIPTNAME
          "/reportApp/parseData.sh ^REP_PATH^" STREAMLOGON salesadm
SC1_PROCESS_DATA SCRIPTNAME
          "/reportApp/processData.sh ^REP_PATH^" STREAMLOGON salesadm
SC1_CREATE_REPORTS SCRIPTNAME
          "/reportApp/createReports.sh ^REP_PATH^" STREAMLOGON salesadm
```

3. Define a single job stream with the following content:
   a. Two run cycles:
      - A weekly run cycle referencing the SC1_WEEKLY_DATA_TABLE variable table
      - A monthly run cycle referencing the SC1_MONTHLY_DATA_TABLE variable table
   b. A dependency on the file containing the data used for report generation.
   c. A variable used in the path of the job stream file dependency.

```
SCHEDULE SC1_RUN_REPORTS
  ON RUNCYCLE SC1_MONTHLY_RCY VARTABLE SC1_MONTHLY_DATA_TABLE
     "FREQ=MONTHLY; INTERVAL=1; BYMONTHDAY=27" AT 0800
  ON RUNCYCLE SC1_WEEKLY_RCY VARTABLE SC1_WEEKLY_DATA_TABLE
     "FREQ=WEEKLY; INTERVAL=1; BYDAY=FR" AT 0900
  OPENS "^REP_PATH^/raw_data.out"
  :
  SC1_PARSE_DATA
  SC1_PROCESS_DATA FOLLOWS SC1_PARSE_DATA
  SC1_CREATE_REPORTS FOLLOWS SC1_PROCESS_DATA
END
```

4. Create a plan of 30 days to generate multiple instances of the job stream by running the following command:

   ```
   JnextPlan -days 30
   ```

### Expected results

The *REP_PATH* variable assumes different values according to the run cycle that applies, that is according to the variable table referenced in the corresponding run cycle.

In this way, the job stream instances have a dependency on a different file according to the type of report they are producing, monthly or weekly. That is:

- The job stream instances that generate the weekly report have a dependency on the file containing the data located in the directory `/reports/weekly`.
- The job stream instances that generate the monthly report have a dependency on the file containing the data located in the directory `/reports/monthly`.

Furthermore, the name of the target directory is correctly replaced in the task string of the three jobs run by every job stream instance as follows:

- The jobs run by job stream instances that generate the weekly report run shell scripts with the `/reports/weekly` directory as an input argument.
- The jobs run by job stream instances that generate the monthly report run shell scripts with the `/reports/monthly` directory as an input argument.

## Customizing jobs and job streams based on why they are scheduled to run

This scenario shows how to customize jobs and job streams based on why they are scheduled to run, for example to create a job that runs different commands for different users, referencing different variable tables at job stream level.

By having the ability to manage different tables of variables, you no longer need to create two different job definitions to run the same workload in different environments. Instead, you define a single job that can be used both in the test and in the production phase, using variables in the name of the file the job runs (**scriptname** job attribute) and in the name of the user under which the job runs (**streamlogon** job attribute). The variable tables are referenced in the job streams containing the job. Using variable tables you reduce the number of scheduling objects required to implement the desired workload saving therefore time and money.

You can find more information on this subject in the *User's Guide and Reference*.

### Scenario goal

After an acquisition, a company needs to integrate the acquired company applications to its existing business environment. Before running them in the production phase, the company decides to try a test application that runs the new company application. After a month of testing, the application of the acquired company will run in the production environment with the production user.

### Roles and skills

This section lists the users involved in the scenario, the related roles, and the required skill level:

**IBM Workload Scheduler Job Scheduler**
Manages Tivoli Workload Scheduler workload. Required skills include Tivoli Workload Scheduler knowledge.

### System requirements

Install the following software before starting the scenario:

- IBM Workload Scheduler version 8.5
- Dynamic Workload Console version 8.5

### Setting up the environment

Complete the following tasks before running the scenario:

- Install and configure IBM Workload Scheduler version 8.5
- Install and configure Dynamic Workload Console version 8.5

## Running the Scenario

To complete the scenario, perform the following steps:

### Procedure

1. Define the TABLE_TEST and the TABLE_PROD variable tables as follows:

```
VARTABLE TABLE_TEST
  MEMBERS
  USER_LOGON "test"
  PATH_SCRIPT "/usr/test"
END

VARTABLE TABLE_PROD
  MEMBERS
  USER_LOGON "mdm_85"
  PATH_SCRIPT "/export/home/mdm_85/TWS"
END
```

2. Define the following job:

```
JOB_APP_01
 SCRIPTNAME ^PATH_SCRIPT/applicationscript.sh"
 STREAMLOGON "^USER_LOGON^"
```

3. Define the following job streams, specifying the same name but different validity intervals:

```
SCHEDULE JS_APP_01
  VARTABLE TABLE_TEST
  ON RUNCYCLE RULE1 "FREQ=DAILY;"
  :
  JOB_APP_01
END

SCHEDULE JS_APP_01
   VALIDFROM 05/19/2008
   VARTABLE TABLE_PROD
   ON RUNCYCLE RULE2 "FREQ=DAILY;"
  :
  JOB_APP_01
END
```

4. Generate a daily plan that ends on 05/18/2008, that is, the last day of the test phase.
5. When you complete the test phase, extend the plan by one day.

### Expected results

After you perform Step 4, the JS_APP_01 job stream will be included into the plan generated, so that the JOB_APP_01 test job is included in the plan and you can verify that the job is correctly scheduled and will run the test application.

After you perform Step 5, the JS_APP_01 job stream is included into the plan running the JOB_APP_01 that uses as **streamlogon** and **scriptname** the values to run the acquired company application in the production environment, without having to modify the scheduling objects.

You can achieve the same result by using a single job stream in which you define two run cycles with different validity intervals, each run cycle referencing a different variable tables. The job stream has the following syntax:

```
SCHEDULE JS_APP_01
   ON RUNCYCLE RULE1 VALIDTO 05/18/2008 VARTABLE TABLE_TEST "FREQ=DAILY;"
   ON RUNCYCLE RULE2 VALIDFROM 05/19/2008 VARTABLE TABLE_PROD "FREQ=DAILY;"
   :
   JOB_APP_001
END
```

## Customizing jobs and job streams based on where they are scheduled to run

This scenario shows how to customize jobs and job streams based on where they run, for example, on different workstations, using variable tables with workstations.

By having the ability to manage different variable tables, you no longer need to create two different job definitions to run the same workload (for example, a command whose syntax changes depending on the operating system) on different workstations. Instead, you define a single job that can be used on different workstations, defining a different variable table for each workstation and referencing it in the workstation definition. Each variable table contains the same variable with different values. You can than use that variable in the command that the job runs. Using variable tables you reduce the number of scheduling objects required to implement the desired workload saving therefore time and money.

You can find more information on this subject in the *User's Guide and Reference*.

### Scenario goal

A company needs to run a clean-up application on the workstations in the sales department, once a week. These workstations have different IBM Workload Scheduler home directories and different IBM Workload Scheduler users. Using variable tables, you define a single job that runs the IBM Workload Scheduler **rmstdlist** clean-up application and a single job stream that includes this job. Your sales department includes multiple workstations which are all members of the same workstation class, so you define a different variable table for each workstation and reference it in the workstation definition. You define a job on the workstation class. This job must contain a variable for the path in the **scriptname** job attribute and a variable in the **streamlogon** job attribute.

### Roles and skills

This section lists the users involved in the scenario, the related roles, and the required skill level:

**IBM Workload Scheduler Job Scheduler**
Manages Tivoli Workload Scheduler workload. Required skills include Tivoli Workload Scheduler knowledge.

### System requirements

Install the following software before starting the scenario:

- IBM Workload Scheduler version 8.5
- Dynamic Workload Console version 8.5

### Setting up the environment

Complete the following tasks before running the scenario:

- Install and configure IBM Workload Scheduler version 8.5
- Install and configure Dynamic Workload Console version 8.5

## Running the Scenario

To complete the scenario, perform the following steps:

### Procedure

1. Define the SC2_PATH_TABLE1 and the SC2_PATH_TABLE2 variable tables as follows:

```
VARTABLE SC2_PATH_TABLE1
  MEMBERS
  TWS_HOME "/home/MDM_85/TWS"
  TWS_USER "mdm_85"
END

VARTABLE SC2_PATH_TABLE2
  MEMBERS
  TWS_HOME "/var/fta_85/TWS"
  TWS_USER "fta_85"
END
\
```

2. Reference the SC2_PATH_TABLE1 and the SC2_PATH_TABLE2 variable tables on the workstations in your department. For example:
   - Reference the SC2_PATH_TABLE1 variable table on the SC2_TWS1 workstation using the following syntax:

```
CPUNAME SC2_TWS1
  DESCRIPTION workstation of sales department
  VARTABLE SC2_PATH_TABLE1
  OS UNIX
  NODE iva.sales.com
  ....
END
```

   - Reference the SC2_PATH_TABLE2 variable table on the SC2_TWS2 workstation using the following syntax:

```
CPUNAME SC2_TWS2
  DESCRIPTION workstation of sales department
  VARTABLE SC2_PATH_TABLE2
```

```
    OS UNIX
    NODE iva.sales.com
    ....
  END
```

3. Create the SC2_TWSWCLASS workstation class containing the SC2_TWS1 and the SC2_TWS2 workstations as members as follows:

```
SC2_TWSWCLASS
   MEMBERS
   SC2_TWS1
   SC2_TWS2
END
```

4. Create the SC2_JOB_CLEANUP job definition that runs the **rmstdlist** clean-up application using the following syntax:

```
SC2_TWSWCLASS#SC2_JOB_CLEANUP
DOCOMMAND "^TWS_HOME^/bin/rmstdlist" STREAMLOGON "TWS_USER"
```

5. Include the SC2_JOB_CLEANUP job definition in the following job stream:

```
SCHEDULE SC2_TWSWCLASS#SC2_JS_CLEANUP
ON SU
:
SC2_TWSCLASS#SC2_JOB_CLEANUP
END
```

### Expected results

When a plan is created on Sunday, one instance of the job stream for each of the workstations in the workstation class is included in the plan. The *TWS_HOME* and the *TWS_USER* variables are resolved differently according to the workstation on which the job is defined.

## Customizing jobs and job streams for submission

This scenario shows how you customize jobs and job streams for submission, for example, to submit a job that runs different data when a specific event occurs in your workload environment.

By having the ability to manage different tables of variables, you no longer need to create multiple job definitions to run different data when a specific event occurs. Instead, you define a single job that can be used when a specific event occurs, using variables in the data that the job runs (**scriptname** job attribute) and specifying the variable table containing the data related to the event when submitting the job. Using variable tables you reduce the number of jobs and job streams required to implement your workload therefore saving time and money.

You can find more information on this subject in the *User's Guide and Reference*.

### Scenario goal

When a new employee is hired, a company wants to run an application to set up the environment for him. To be installed this application requires information such as the name of the employee, the department to which he belongs, and so on. To address this need the company performs the following actions:

- Defines a variable table that contains the variables that identify the new employee.
- Defines a unique job containing the variables required to install the application to be used for any new employee.

- Specify the variable table that contains the variables that identify the new employee.

### Roles and skills

This section lists the users involved in the scenario, the related roles, and the required skill level:

**IBM Workload Scheduler Job Scheduler**
Manages IBM Workload Scheduler workload. Required skills include Tivoli Workload Scheduler knowledge to:
- Manage workload complexity and dependencies.
- Optimize schedule efficiency, flexibility, and resiliency.

**IBM Workload Scheduler Operator**
Manages all operational processes and procedures, ensuring the business continuity of the workflow. Required skills include Tivoli Workload Scheduler knowledge to:
- Monitor critical events and perform first analysis of problems.
- Manage and coordinate the resolution of issues

### System requirements

Install the following software before starting the scenario:
- IBM Workload Scheduler version 8.5
- Dynamic Workload Console version 8.5

### Setting up the environment

Complete the following tasks before running the scenario:
- Install and configure IBM Workload Scheduler version 8.5
- Install and configure Dynamic Workload Console version 8.5

## Running the Scenario

To complete the scenario, perform the following steps:

### Procedure

1. Define the JOB_SETUP_HR job used to install the application that sets up the environment for any new employee as follows:

```
JOB_SETUP_HR
 SCRIPTNAME "/var/setupHRApp ^HOST^ ^PORT^ ^USER^ ^DEPT^"
 STREAMLOGON admin
```

2. Define the JH_TABLE_ID001 new employee variable table as follows:

```
 VARTABLE JH_TABLE_ID001
   MEMBERS
   HOST "cpux.acme.com"
   PORT "42577"
   USER "Jennifer Harold"
   DEPT "Sales"
 END
```

3. Submit the JOB_SETUP_HR job, specifying the JH_TABLE_ID001 new employee variable table. To do this, run the following command:

```
conman sbj JOB_SETUP_HR ;vt=JH_TABLE_ID001
```

## Expected results

After you submit the JOB_SETUP_HR job, the company environment for the new employee is set up. You can reuse the same job for any new employee just by specifying the variable table containing the new employee data when you submit the job.

# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing*
*Legal and Intellectual Property Law*
*IBM Japan Ltd.*
*19-21, Nihonbashi-Hakozakicho, Chuo-ku*
*Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM® Corp. and Quantum in the U.S. and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.


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

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

ITIL is a Registered Trade Mark of AXELOS Limited.

UNIX is a registered trademark of The Open Group in the United States and other countries.

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

## Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

## Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

## Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

## Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

# Index

## S

Product Number: 5698-WSH

Printed in USA