Scheduling Services

# Overview

Use the server's scheduling function to schedule services to execute at times you specify. Services that you schedule are referred to as **user tasks**. You can:

* Create user tasks that Integration Server executes:
  + A single time
  + Repeatedly at a simple interval (e.g., hourly every day)
  + Repeatedly at a complex interval (e.g., every other Tuesday in June)
* View a list of scheduled tasks
* Update scheduling options for existing user tasks
* Cancel a scheduled user task before Integration Server completes all scheduled executions
* Temporarily suspend a task's execution
* Create user tasks that run on one, any, or all Integration Servers connected to the same database
* Specify an action Integration Server is to take if a task has missed its scheduled execution time

**Note**: In addition to using Integration Server Administrator to schedule tasks, you can also perform scheduling by using a set of built-in services. See the *webMethods Integration Server Built-In Services Reference* for more information.

**Tasks Provided by Integration Server**

Integration Server provides user tasks that you can modify.  
Example: Integration Server supplies the **Message History Sweeper** task if you configured a document history database for exactly-once processing. This task removes expired entries from the document history database. Even though Integration Server scheduled this task, you can modify how often the service runs.

In addition to the scheduled user tasks that you set up, Integration Server schedules **system tasks** that it performs for normal system operation. You can view, but not update or cancel, most scheduled system tasks.

# **two main ways** to create scheduled tasks:

In webMethods Integration Server, there are **two main ways** to create scheduled tasks:

**🧩 1. Using Integration Server Administrator (IS Admin Page)**

* Navigate to **Server > Scheduling**
* Use the UI to:
  + Create one-time, repeating, or complex tasks
  + Assign inputs, set run times, and configure late actions
  + Choose target nodes and user credentials
* Ideal for **manual setup and quick configuration**

**🧪 2. Using webMethods Designer (via Flow Services)**

* Use built-in services from the **pub.scheduler folder**:
  + addOneTimeTask, addRepeatingTask, addComplexTask
  + suspendTask, resumeTask, cancelTask, etc.
* Allows **dynamic scheduling** from within your logic
* Ideal for **automated or conditional scheduling** based on business rules

**🧠 When to Use Which?**

| **Method** | **Best For** |
| --- | --- |
| **IS Admin Page** | Manual setup, testing, admin control |
| **Designer + Flow Services** | Programmatic scheduling, dynamic logic, automation |

## 1.Scheduling a User Task (IS Admin Page)

**Procedure**

1. Open the Integration Server Administrator.
2. Go to **Server > Scheduling**.
3. Click **Create a Scheduled Task**.
4. Set the **Service Information** parameters:

| **Parameter** | **Description** |
| --- | --- |
| Description | A description of the task |
| folder.subfolder:service | The fully qualified service name of the service you want Integration Server to execute |

Click **Assign Inputs** to enter new input values or modify existing values for the service you want Integration Server to execute.

**Notes**:

* You can only assign values for top-level input parameters of data type String.
* Assigning input values using Assign Inputs overwrites any input values assigned to a service using the built-in services in the pub.scheduler folder.
* For information about specifying service names, see *Fully Qualified Service Names*.

**Run As User**

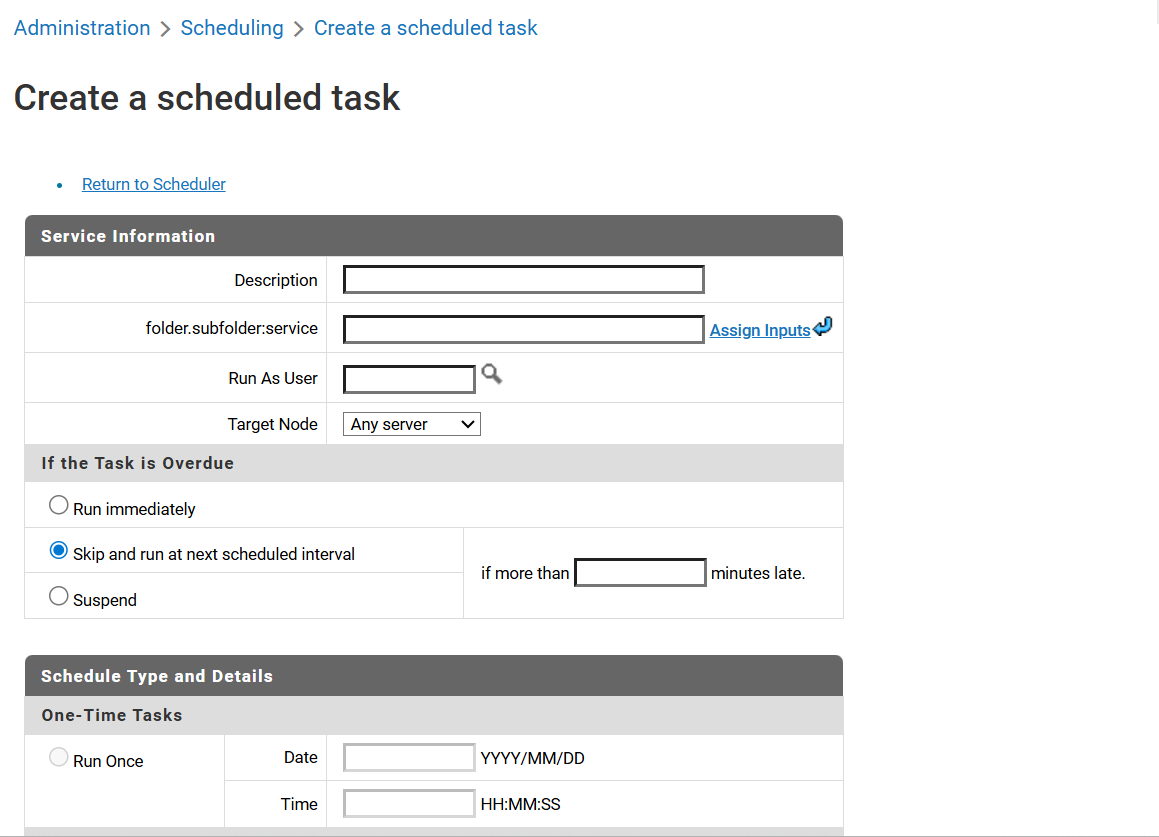
* Specify the user name you want the server to use when running the service.
* A user can be selected from the local or central directory.
* Integration Server runs the service as if the user you specify is the authenticated user that invoked the service.
* If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.

**Target Node**

Specify where the task should run:

* **Any server**: Run on only one server connected to the database (random selection).
* **All servers**: Run on all Integration Servers in a stateful cluster. Not available for stateless clusters.
* **Specific server**: Select the name from the list of Integration Servers connected to the database.
* Default is the current Integration Server.

For more information, see *Target Node Options*.



**If the Task Is Overdue**

Specify what Integration Server should do if a task misses its scheduled execution time:

| **Option** | **Behavior** |
| --- | --- |
| Run immediately | Run the task immediately, no matter how late it is |
| Skip and run at next scheduled time | Skip this execution and run it again at the next scheduled time (not available for one-time tasks) |
| Suspend | Place the task in a suspended state until resumed or canceled by an administrator |

Integration Server periodically checks the status of scheduled tasks. If it finds a task that should have started but has not, it runs the task immediately unless you specify a special action for late tasks. Integration Server performs this "late action" if the task has missed its scheduled start time by the number of minutes you specify in the **if more than xxx minutes late** field.

**Note**: The maximum value for the **if more than xxx minutes late** field is 35,000 minutes.  
For tasks that are late but do not exceed the specified period, Integration Server runs the task immediately.

**Scheduling Options**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Run Once**

Schedule a task to run just once. Specify:

* **Date**: Format YYYY/MM/DD (e.g., 2010/03/11)
* **Time**: Format HH:MM:SS (24-hour clock, e.g., 13:00:00 for 1:00 PM)

**Repeating**

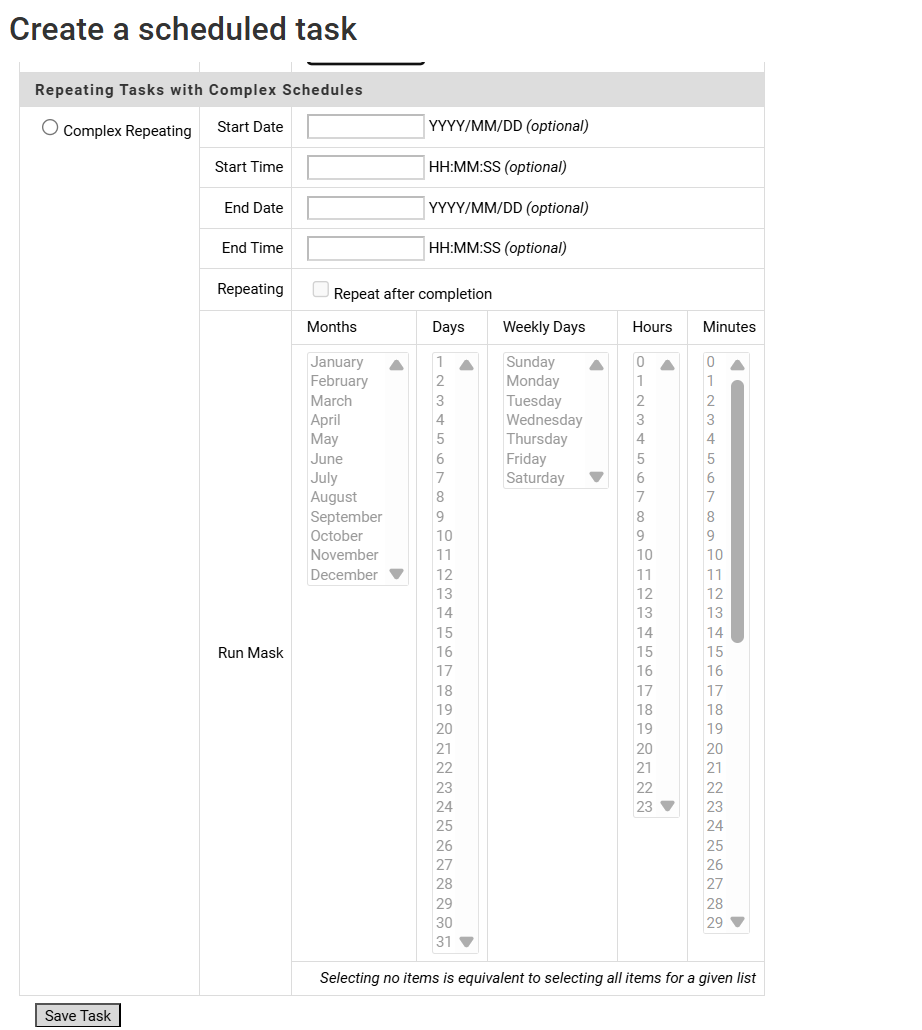
Schedule simple repeating tasks. Specify:

* **Start Date and Start Time**: Format YYYY/MM/DD and HH:MM:SS
* **End Date and End Time**: Optional; same format
* **Interval**: Number of seconds between executions
* **Repeat after completion**: Whether to wait for the previous execution to complete before starting the next

**Example**:  
If GetData runs every minute but takes longer than a minute to complete, checking this box ensures the next run waits until the previous one finishes.

For more information, see *Simple Repeating Option*.

**Complex Repeating**

****

Schedule tasks that repeat at complex intervals (e.g., every other day, twice a month). Specify:

* **Start Date and Start Time**: Optional; uses Run Mask if omitted
* **End Date and End Time**: Optional; uses current date/time if omitted
* **Run Mask**: Select specific months, dates (1–31), days of the week (Sunday–Monday), hours, minutes, and seconds.

Use Ctrl to select multiple items. If no items are selected, all are assumed.

* **Repeat after completion**: Whether to wait for the previous execution to complete before starting the next

**Example**:  
If GetData runs every minute but takes longer than a minute to complete, checking this box ensures the next run waits until the previous one finishes.

For more information, see *Complex Repeating Option*.

**🔁 Simple Repeating Tasks**

These run at fixed intervals (e.g., every 15 minutes).

| **Configuration** | **Behavior** |
| --- | --- |
| **Start Date/Time omitted** | Task starts **immediately** at the time of creation |
| **End Date/Time omitted** | Task runs **indefinitely** until manually canceled |
| **Both omitted** | Task starts now and runs forever at the defined interval |

**🔄 Complex Repeating Tasks**

These run based on a **Run Mask** (e.g., every Monday at 6:00 AM).

| **Configuration** | **Behavior** |
| --- | --- |
| **Start Date/Time omitted** | Task starts at the **next matching Run Mask time** |
| **End Date/Time omitted** | Task runs **only today**, if today matches the Run Mask; otherwise, it **does not run again** |
| **Both omitted** | Task runs **indefinitely** on all matching Run Mask dates and times |

**✅ Example for Complex Repeating**

* **Run Mask**: Every Monday at 6:00 AM
* **Today**: Thursday
* **Start Date/Time provided**, **End Date/Time omitted**  
  → Task does **not run**, because today is not Monday  
  → Task is **discarded after today**

To make it run every Monday forever:  
→ **Omit both Start and End Date/Time**

## 2.Scheduling a User Task (pub.scheduler)

The **pub.scheduler folder** in webMethods Integration Server contains built-in services that allow you to programmatically manage scheduled tasks. These services are part of the **WmPublic package** and provide full control over creating, updating, querying, and managing task execution.

**📂 Overview of pub.scheduler Folder**

This folder is used to:

* Schedule services to run once, repeatedly, or on complex intervals
* Suspend, resume, or cancel tasks
* Retrieve task details and lists
* Migrate tasks to external databases

**🧰 Services in the Folder**

| **Service Name** | **Description** |
| --- | --- |
| addComplexTask | Adds a task with complex scheduling (e.g., specific days, times, months) |
| addOneTimeTask | Adds a task that runs only once at a specified date and time |
| addRepeatingTask | Adds a task that runs at regular intervals (e.g., every 15 minutes) |
| cancelTask | Removes a scheduled task from the Scheduler |
| getTaskIDs | Retrieves a list of task IDs currently in the Scheduler |
| getTaskInfo | Retrieves detailed information about a specific scheduled task |
| getUserTaskList | Returns a list of all user-defined scheduled tasks |
| migrateTasksToJDBC | Migrates scheduled tasks from the embedded database to an external JDBC database |
| resumeTask | Resumes a suspended task so it can continue executing |
| suspendTask | Suspends a task, preventing it from running until resumed |
| updateComplexTask | Updates the configuration of an existing complex task |
| updateOneTimeTask | Updates the configuration of an existing one-time task |
| updateRepeatingTask | Updates the configuration of an existing repeating task |

**🧠 Use Cases**

* Automate data syncs, cleanups, or report generation
* Dynamically schedule tasks from within flow services
* Manage task execution across clustered environments
* Monitor and audit scheduled task behavior