

**ActiveBatch: An Overview**

By Randy Fadler

December 2024.

ActiveBatch is a comprehensive workload automation and job scheduling solution designed to streamline IT operations and enhance efficiency across various industries. With its intuitive low-code/no-code interface, ActiveBatch empowers users to automate complex workflows without extensive coding knowledge, making it accessible to both technical and non-technical users. The platform offers advanced job scheduling, event-driven automation, and a vast library of pre-built job steps, simplifying the creation and management of automated processes. ActiveBatch's robust integration capabilities support a wide range of applications, databases, and platforms, ensuring seamless end-to-end automation. Additionally, the platform provides real-time monitoring, customizable alerts, and detailed reporting tools, offering complete visibility and control over job statuses and service level agreements (SLAs). Security is a top priority, with features such as multi-factor authentication, granular permissions, and data encryption ensuring the protection of automated processes and data. ActiveBatch also supports high availability configurations, ensuring continuous operation and minimizing downtime. With its user-friendly design, extensive integration options, and powerful automation capabilities, ActiveBatch is an invaluable tool for organizations looking to optimize their IT operations and drive business success.

Contents

[Overview of ActiveBatch 4](#_Toc185147373)

[Key Features 4](#_Toc185147374)

[Examples of Use Cases 4](#_Toc185147375)

[Pros and Cons of Using ActiveBatch 5](#_Toc185147376)

[ActiveBatch vs Autosys 5](#_Toc185147377)

[Companies Using ActiveBatch 6](#_Toc185147378)

[Job Scheduling 7](#_Toc185147379)

[ActiveBatch Variables 9](#_Toc185147380)

[Built-in Variables 10](#_Toc185147381)

[ActiveBatch Console 11](#_Toc185147382)

[Key Components of the ActiveBatch Console 11](#_Toc185147383)

[Features of the ActiveBatch Console 12](#_Toc185147384)

[ActiveBatch Job and Plan Monitoring 12](#_Toc185147385)

[ActiveBatch Job Step Library 13](#_Toc185147386)

[Using the ActiveBatch Console 15](#_Toc185147387)

[Requirements For Running ActiveBatch 16](#_Toc185147388)

[Supported Programming Languages 18](#_Toc185147389)

[Script Optimization and Management 19](#_Toc185147390)

[ActiveBatch Security 19](#_Toc185147391)

[Circular Loop Protection 20](#_Toc185147392)

[ActiveBatch Terminology 21](#_Toc185147393)

[Using Conditionals on Jobs and Plans 23](#_Toc185147394)

[Advanced Features 24](#_Toc185147395)

[Resources, Alerts, and References 25](#_Toc185147396)

[Reporting Services 27](#_Toc185147397)

[Conclusion 28](#_Toc185147398)

# Overview of ActiveBatch

ActiveBatch is a comprehensive IT automation solution designed to streamline and automate complex workflows across various platforms and applications. It offers a centralized control point for managing and monitoring automated processes, making it an essential tool for IT operations, DevOps, and business process automation.

## Key Features

1. **Job Scheduling and Automation**: ActiveBatch allows you to create, schedule, and manage jobs and workflows with ease. It supports various triggers, including time-based, event-based, and conditional triggers, ensuring that tasks are executed precisely when needed.
2. **Integration Capabilities**: ActiveBatch integrates seamlessly with a wide range of applications, databases, and platforms. This includes ERP systems, cloud services, and custom applications, enabling you to automate end-to-end processes across your entire IT environment.
3. **Low-Code/No-Code Interface**: The platform features a user-friendly, drag-and-drop interface that allows users to build and modify workflows without extensive coding knowledge. This makes it accessible to both technical and non-technical users.
4. **Self-Service Portal**: ActiveBatch offers a self-service portal that empowers business users and help desk teams to initiate and manage processes without IT intervention. This reduces the workload on IT teams and speeds up response times.
5. **Advanced Monitoring and Reporting**: The tool provides real-time monitoring and detailed reporting capabilities. Users can track the status of jobs, receive alerts for failures or delays, and generate comprehensive reports for analysis and auditing.

## Examples of Use Cases

1. **Data Integration and ETL Processes**:
	* **Scenario**: A company needs to extract data from multiple sources, transform it, and load it into a data warehouse.
	* **Solution**: Using ActiveBatch, you can create a workflow that automates the entire ETL process. The workflow can be triggered by specific events, such as the arrival of new data files, and can include error handling and notifications to ensure data integrity and timely completion.
2. **Cloud Resource Management**:
	* **Scenario**: An organization wants to optimize its cloud resource usage by automatically scaling resources based on demand.
	* **Solution**: ActiveBatch can integrate with cloud platforms like AWS and Azure to automate the provisioning and de-provisioning of resources. Workflows can be set up to monitor resource usage and trigger scaling actions based on predefined thresholds.
3. **Batch Processing for Financial Services**:
	* **Scenario**: A financial institution needs to run nightly batch processes to update account balances, generate reports, and perform data backups.
	* **Solution**: ActiveBatch can schedule and manage these batch processes, ensuring they run in the correct sequence and handling any dependencies between tasks. The platform's monitoring capabilities can alert IT staff to any issues, allowing for quick resolution.

## Pros and Cons of Using ActiveBatch

**Pros**:

* **Versatile Integration**: Supports a wide range of applications and platforms.
* **User-Friendly Interface**: Low-code/no-code environment makes it accessible to a broad audience.
* **Robust Monitoring**: Real-time tracking and detailed reporting enhance visibility and control.

**Cons**:

* **Complexity**: The extensive features can be overwhelming for new users, and the learning curve can be steep.
* **Cost**: ActiveBatch can be expensive, particularly for smaller organizations.

**Wrap-up**

ActiveBatch is a powerful and versatile tool for automating and managing complex workflows. Its integration capabilities, user-friendly interface, and robust monitoring features make it a valuable asset for organizations looking to streamline their IT operations and improve efficiency.

## ActiveBatch vs Autosys

**Feature Comparison**

1. **Ease of Use**:
	* **ActiveBatch**: Known for its user-friendly interface with low-code/no-code functionality, making it accessible for both technical and non-technical users.
	* **AutoSys**: Also, user-friendly but often praised for its simplicity and ease of setup.
2. **Integration Capabilities**:
	* **ActiveBatch**: Offers extensive integration capabilities with a wide range of applications, databases, and platforms. It includes pre-built connectors and a REST API for seamless integration.
	* **AutoSys**: Provides robust integration features but may require more custom scripting compared to ActiveBatch.
3. **Scalability**:
	* **ActiveBatch**: Highly scalable, suitable for large enterprises with complex workflows.
	* **AutoSys**: Also highly scalable, often used in large organizations for enterprise-level job scheduling.
4. **Monitoring and Reporting**:
	* **ActiveBatch**: Offers advanced monitoring and reporting features, including real-time tracking, customizable alerts, and detailed reports.
	* **AutoSys**: Provides strong monitoring and reporting capabilities, but some users desire improvements in these areas.
5. **Support and Documentation**:
	* **ActiveBatch**: Users generally find technical support helpful and reliable, with a clear knowledge base and APIs.
	* **AutoSys**: Also receives high praise for its support, with users finding the support team responsive and capable.

**User Reviews and Ratings**

1. **Overall Satisfaction**:
	* **ActiveBatch**: Rated highly for its versatility, ease of use, and integration capabilities. Users appreciate its pre-built jobs and real-time monitoring.
	* **AutoSys**: Valued for its scalability, speed, and availability. Users find it particularly effective for job scheduling and orchestrating tasks.
2. **Deployment and Setup**:
	* **ActiveBatch**: The setup process is generally straightforward, though some users find configuring it on varied environments challenging.
	* **AutoSys**: Known for its effortless and rapid setup, with minimal clicks required for basic configurations.
3. **Pricing**:
	* **ActiveBatch**: Users find the pricing fair and competitive, with a straightforward setup process.
	* **AutoSys**: Pricing details are less frequently discussed, but it is generally considered competitive in the market.

Both ActiveBatch and AutoSys are powerful workload automation tools with their own strengths. ActiveBatch excels in integration capabilities, user-friendly interface, and advanced monitoring features. AutoSys is praised for its scalability, ease of setup, and effective job scheduling.

Choosing between the two will depend on your specific needs, such as the complexity of your workflows, integration requirements, and budget considerations.

## Companies Using ActiveBatch

ActiveBatch is used by a variety of companies across different industries to streamline their IT operations and automate workflows. Here are some notable companies that use ActiveBatch:

1. **Visa**: A global leader in digital payments, Visa uses ActiveBatch to manage and automate its complex IT processes.
2. **T-Mobile**: One of the largest telecommunications companies in the United States, T-Mobile leverages ActiveBatch for efficient job scheduling and automation.
3. **Micron Technology**: A major player in the semiconductor industry, Micron Technology utilizes ActiveBatch to optimize its IT operations.
4. **Providence Health & Services**: A healthcare organization that uses ActiveBatch to automate various administrative and clinical processes.
5. **Infosys**: A global leader in consulting, technology, and outsourcing solutions, Infosys employs ActiveBatch to enhance its IT service management.
6. **Deloitte**: One of the "Big Four" accounting firms, Deloitte uses ActiveBatch to streamline its business processes and improve efficiency.
7. **Capgemini**: A multinational corporation that provides consulting, technology, and outsourcing services, Capgemini integrates ActiveBatch into its IT infrastructure.

These companies represent a diverse range of industries, showcasing the versatility and effectiveness of ActiveBatch in automating and managing complex workflows.

## Job Scheduling

Each ***job*** in ActiveBatch can contain ***one or more job steps***, which are individual tasks or actions that the job performs. Moreover, one job step can produce or otherwise provide the data for a second job. The Job Steps Library in ActiveBatch provides a vast collection of pre-built, production-ready job steps that can be easily assembled into workflows without the need for custom scripting. This library includes job steps for various tasks such as file management, data processing, and system integration, enabling users to automate complex workflows seamlessly. Each pre-built job step has it’s own particular parameters that will be used to accomplish the task. For example, when copying a file, both the source and destination file names must be supplied.

The Job object acts like a ‘program’ in which the memory is shared among all the job steps within the job. So when one job step ends, a second job can perform logic based upon the status of the prior job, or a job could provide a record set which is then consumed by a second job step in the job. This allows the user to build ‘scripts’ without writing any code.

**Example 1: Simple Job Scheduling**

**Scenario**: You need to run a data backup job every night at 2 AM.

**Activities**:

1. **Create a New Job**: In the ActiveBatch console, create a new job and name it "Nightly Data Backup."
2. **Set the Schedule**: Go to the "Schedule" tab and set the job to run daily at 2 AM.
3. **Configure the Job Steps**: Add the necessary steps from the Job Steps Library to perform the data backup, such as **connecting to the database**, **copying files**, and **verifying the backup**.
4. **Save and Activate**: Save the job and activate it. The job will now run automatically every night at 2 AM.

**Example 2: Conditional Job Scheduling**

**Scenario**: You want to run a report generation job only if a specific file is present in a directory.

**Activities**:

1. **Create a New Job**: Create a new job and name it "Conditional Report Generation."
2. **Add a File Trigger**: In the "Triggers" tab, add a file trigger that monitors the directory for the presence of the specific file.
3. **Configure the Job Steps**: Add the steps to generate the report, such as reading data from the file, processing it, and saving the report.
4. **Set Conditions**: In the "Conditions" tab, set the condition to run the job only if the file exists.
5. **Save and Activate**: Save the job and activate it. The job will run automatically when the file is detected.

**Example 3: Workflow Automation**

**Scenario**: You need to automate a workflow that involves multiple jobs with dependencies.

**Activities**:

1. **Create a New Workflow**: In the ActiveBatch console, create a new workflow and name it "Monthly Financial Processing."
2. **Add Jobs to the Workflow**: Add the individual jobs to the workflow, such as data extraction, data transformation, and report generation.
3. **Set Job Dependencies**: Define the dependencies between the jobs, ensuring that each job runs in the correct sequence.
4. **Set the Schedule**: Go to the "Schedule" tab and set the workflow to run on the first day of every month.
5. **Save and Activate**: Save the workflow and activate it. The workflow will run automatically according to the schedule, executing each job in the defined sequence.

**Example 4: Event-Driven Scheduling**

**Scenario**: You want to trigger a job based on an event, such as the completion of another job.

**Activities**:

1. **Create a New Job**: Create a new job and name it "Post-Processing Job."
2. **Add an Event Trigger**: In the "Triggers" tab, add an event trigger that monitors the completion of the preceding job.
3. **Configure the Job Steps**: Add the steps to perform the post-processing tasks, such as data validation and cleanup.
4. **Save and Activate**: Save the job and activate it. The job will run automatically when the preceding job completes.

## ActiveBatch Variables

Global variables in ActiveBatch are used to store values that can be accessed and utilized across multiple jobs and plans. They play a crucial role in passing data between related jobs and plans, and in soft-coding object properties to ensure flexibility and reusability. A variable’s value is gathered by a running job in the following order:

Within the current job, it searches:

1. Job’s queue object,
2. user account object,
3. the job itself,
4. parent container (folder or plan), and the parent’s parent container etc., until the Scheduler root is reached.

 Here are some key points about global variables in ActiveBatch:

**Types of Variables**

1. **Constant Variables**: These maintain a fixed value unless manually changed. They can be strings, integers, doubles, or references to other variables.
2. **Active Variables**: These are derived from built-in data sources and are read-only. Their values may vary from one job run to the next, but they are resolved before the job begins execution.
3. **Execution Variables**: These are declared at execution time, typically within a job step, and are used during the job's runtime.

**Uses of Variables**

* **Data Passing**: Global variables can pass data from one job to another, ensuring seamless data flow within workflows.
* **Variable Substitution**: They can be used to substitute values within plans or jobs, allowing for dynamic and flexible configurations.
* **Constraints**: Variables can act as constraints, where a job or plan will only run if a variable meets a specific condition.
* **Environment Variables**: They can be exported as environmental variables, making their names and values available at execution time.

**Defining Variables**

Global variables are defined in the "Variables" section of an object definition and are evaluated during the pre-processing phase of a plan or job. They can be created at various levels, including:

* Job Scheduler root
* Folders
* Plans
* Jobs
* References (Job and Plan)
* Queues

## Built-in Variables

ActiveBatch also provides several built-in variables that describe various ActiveBatch objects. These variables are prefixed by special characters like $, @, or # and are reserved for ActiveBatch use

Built-in variables in ActiveBatch are predefined variables that provide essential information about the job, plan, or system environment. These variables are automatically available and can be used to enhance the flexibility and functionality of your workflows. Here are some key aspects of built-in variables:

**Types of Built-in Variables**

1. **System Variables**: These provide information about the system environment, such as the current date and time, user information, and system paths.
2. **Job Variables**: These offer details about the job execution, including job name, job ID, job status, and execution time.
3. **Plan Variables**: These give information about the plan, such as plan name, plan ID, and plan status.
4. **Queue Variables**: These provide details about the job queue, including queue name and queue status.

**Common Built-in Variables**

* **$JobName**: The name of the job being executed.
* **$JobID**: The unique identifier for the job.
* **$JobStatus**: The status of the job (e.g., running, completed, failed).
* **$ExecutionTime**: The time at which the job started execution.
* **$PlanName**: The name of the plan being executed.
* **$PlanID**: The unique identifier for the plan.
* **$QueueName**: The name of the queue in which the job is running.
* **$CurrentDate**: The current system date.
* **$CurrentTime**: The current system time.
* **$UserName**: The name of the user executing the job.

**Usage**

Built-in variables can be used in various ways to enhance your workflows:

* **Dynamic Job Names**: Use variables like $JobName to dynamically set job names based on execution context.
* **Conditional Logic**: Implement conditional logic based on job status or execution time using variables like $JobStatus and $ExecutionTime.
* **Logging and Reporting**: Include built-in variables in log files or reports to provide detailed information about job execution.
* **Parameter Passing**: Pass built-in variables as parameters to other jobs or scripts to maintain context and continuity.

These variables are particularly useful for creating dynamic and adaptable workflows, ensuring that your automation processes are both efficient and responsive to changing conditions.

## ActiveBatch Console

The ActiveBatch Console is the **primary interface** for managing and monitoring your automated workflows and job schedules. It provides a comprehensive and user-friendly environment for creating, scheduling, and managing jobs. Let's break down its key components and features:

## Key Components of the ActiveBatch Console

1. **Start Page**:
	* The Start Page is the initial screen you see when you open the ActiveBatch Console. It provides quick access to recent connections, new connections, and various resources such as online help and license management.
2. **Menu Bar**:
	* Located at the top of the console, the Menu Bar provides access to various functions and settings. It includes options for file management, view customization, tools, and help.
3. **Object Navigation Pane**:
	* This pane, often referred to as the "tree," is where users navigate through different objects such as jobs, plans, queues, and alerts. It allows you to add, modify, or take action against these objects.
4. **Views Pane**:
	* The Views Pane includes tabs for Views, Properties, Favorites, Variable Explorer, Tags, and Templates. These tabs allow you to manage and customize different aspects of your jobs and workflows.
5. **Main View**:
	* The Main View is the central area where detailed information about selected objects is displayed. It includes tabs for different views, such as object properties, dependent objects, search results, and more.
6. **Instances Pane**:
	* This pane includes tabs for Instances, Alerts, Instance Details, Generic Queue Instances, and Log Viewer. It provides detailed information about job instances and allows you to monitor and manage them effectively.

## Features of the ActiveBatch Console

1. **Integrated Jobs Library**:
	* The Integrated Jobs Library provides hundreds of pre-built job steps for various systems and platforms. This library simplifies the creation of workflows by eliminating the need for custom scripting.
2. **Service Library**:
	* The Service Library offers production-ready job steps and an intuitive graphical development environment. It helps streamline the development and deployment of automated processes.
3. **Event-Driven and Time-Based Triggers**:
	* ActiveBatch supports both event-driven and time-based triggers, allowing you to automate processes based on specific events or schedules. This flexibility helps ensure that jobs are executed precisely when needed.
4. **SLA Management**:
	* The console includes robust SLA management capabilities, allowing you to monitor and manage critical workflows to meet service level agreements. You can visualize key details, customize alerts, and automate remediation to ensure optimal service delivery.
5. **Reporting and Monitoring**:
	* ActiveBatch provides comprehensive monitoring, alerting, and reporting features. You can track job statuses, receive alerts for failures or delays, and generate detailed reports for analysis and auditing.
6. **Security and Governance**:
	* The console includes robust security features, such as authorized access and granular permissions. It ensures that security is maintained across on-premises, cloud, and hybrid environments.

## ActiveBatch Job and Plan Monitoring

ActiveBatch offers robust job and plan monitoring capabilities to ensure smooth and efficient workflow execution. Here are some key features and aspects of job and plan monitoring in ActiveBatch:

**Monitoring Capabilities**

1. **Real-Time Monitoring**: ActiveBatch provides real-time monitoring through the console of jobs and plans, allowing you to track the progress and status of your workflows as they execute.
2. **Customizable Alerts**: You can set up customizable alerts to notify you of specific events, such as job failures, delays, or SLA breaches. Alerts can be delivered via email, SMS, JMS, Microsoft Message Queues, Twitter, SNMP, Skype, and more.
3. **Proactive Monitoring**: ActiveBatch proactively monitors workflow progress and takes automated steps if a workflow overruns, underruns, fails, or is in danger of breaching an SLA.
4. **Multiple Views**: ActiveBatch provides multiple views for both developers and operations, including Machine Load View, Forecast View, and Gantt View. These views help you analyze data, predict optimal run times, and manage system usage.

**Alerting and Remediation**

* **Automated Remediation**: ActiveBatch supports automated remediation actions based on alert conditions. Workflows can take different branches based on the conditions raised, ensuring seamless recovery from issues.
* **Custom Messages**: You can customize alert messages and subjects using ActiveBatch Variables, making it easier to understand and respond to alerts.
* **SLA Management**: ActiveBatch allows you to set warning and critical thresholds for SLAs and establish customized alert notifications and automated remediation actions for impending SLA breaches.

**Reporting**

* **Detailed Reporting**: ActiveBatch provides detailed reporting tools that offer complete visibility and control over job statuses and SLAs. Reports can be customized to meet specific needs and provide insights into workflow performance.

These features make ActiveBatch a powerful tool for monitoring and managing your automated workflows, ensuring that any issues are quickly identified and addressed to maintain optimal performance.

## ActiveBatch Job Step Library

ActiveBatch includes a comprehensive Job Steps Library with hundreds of pre-built job steps designed to simplify and streamline automation across various systems and platforms. Here are some examples of the built-in libraries and the types of job steps they offer:

**Examples of ActiveBatch Built-In Libraries**

1. **File System Job Steps**:
	* **File Management**: Copy, move, rename, and delete files and directories on target operating systems.
	* **File Transfer**: Upload and download files using FTP, SFTP, and FTPS protocols.
2. **Database Job Steps**:
	* **SQL Execution**: Run SQL queries and stored procedures on databases like SQL Server, Oracle, and MySQL.
	* **Database Maintenance**: Perform tasks such as backups, restores, and index maintenance.
3. **Managed File Transfer (MFT) Job Steps**:
	* **File Upload/Download**: Securely transfer files between systems using various protocols.
	* **File Encryption/Decryption**: Encrypt and decrypt files to ensure data security during transfer.
4. **Cloud Integration Job Steps**:
	* **AWS**: Automate tasks such as EC2 instance management, S3 bucket operations, and Lambda function execution.
	* **Azure**: Manage Azure resources, including virtual machines, storage accounts, and Azure Functions.
5. **ERP and Business Systems Job Steps**:
	* **SAP**: Execute SAP jobs, manage SAP data, and integrate with SAP applications.
	* **Oracle E-Business Suite**: Automate Oracle EBS processes and data management tasks.
6. **Reporting and BI Job Steps**:
	* **Crystal Reports**: Generate and distribute Crystal Reports.
	* **Cognos**: Run Cognos reports and manage Cognos data.
7. **IT Service Management (ITSM) Job Steps**:
	* **ServiceNow**: Create, update, and manage ServiceNow incidents, changes, and tasks.
	* **BMC Remedy**: Integrate with BMC Remedy for IT service management automation.
8. **Virtualization and Cloud Management Job Steps**:
	* **VMware**: Manage VMware virtual machines, snapshots, and other resources.
	* **Hyper-V**: Automate Hyper-V virtual machine operations and management tasks.
9. **Big Data and Analytics Job Steps**:
	* **Hadoop**: Manage Hadoop jobs, data transfers, and cluster operations.
	* **Informatica**: Integrate with Informatica PowerCenter and Informatica Cloud for data integration tasks.
10. **Custom Job Steps**:
	* **REST API**: Create custom job steps using REST APIs to integrate with virtually any application or service.
	* **.NET Assemblies**: Develop custom job steps using .NET assemblies for specific automation needs.

These built-in libraries provide a script-less approach to building jobs and workflows, allowing users to automate a wide range of tasks without extensive coding

## Using the ActiveBatch Console

The console's layout is designed to provide a clear and organized view of your automated processes, making it easy to manage and monitor your workflows. By leveraging the various panes and features, you can efficiently create, schedule, and manage jobs to streamline your IT operations.

**Getting Started with the ActiveBatch Console**

1. **Launching the Console**:
	* You can start the ActiveBatch Console by double-clicking its desktop icon or selecting it from the ActiveBatch program menu.
	* Upon launching, you'll see a splash screen displaying the version and registered owner information.
2. **Connecting to a Job Scheduler**:
	* The first time you run the console, it will display the Start Page. Here, you can establish a new connection to a Job Scheduler by clicking on the "New Connection" link.
	* Enter the hostname of the Job Scheduler and click "Connect" or "Test" to ensure the connection is successful.
3. **Console Layout**:
	* The console is divided into several key areas:
		+ **Navigation Pane**: This is where you can browse through different objects and instances.
		+ **View Pane**: Displays detailed information about the selected object.
		+ **Main Pane**: The primary workspace where you can create and manage jobs.
		+ **Instances Pane**: Shows the status and history of job instances.

**Creating and Managing Jobs**

1. **Creating a Job**:
	* To create a new job, navigate to the appropriate folder in the Navigation Pane and select "New Job".
	* Fill in the job details, such as name, description, and job type. You can choose from various job types, including script jobs, file transfer jobs, and more.
2. **Configuring Job Steps**:
	* Jobs can consist of multiple steps. Use the Integrated Jobs Library to drag and drop pre-built job steps into your job workflow.
	* Configure each step by specifying parameters, conditions, and dependencies.
3. **Scheduling Jobs**:
	* Set the job's schedule by defining triggers and conditions. You can schedule jobs to run at specific times, on certain events, or based on dependencies.
	* Use the Calendar View to visualize and manage job schedules.

**Monitoring and Reporting**

1. **Real-Time Monitoring**:
	* The console provides real-time monitoring of job statuses. You can view job progress, logs, and any errors that occur during execution.
	* Set up customizable alerts to notify you of job statuses and issues.
2. **Reporting**:
	* Generate detailed reports on job performance, execution history, and resource utilization.
	* Use these reports to analyze and optimize your workflows.

**Security and Permissions**

1. **User Management**:
	* Manage user accounts and permissions to control access to the ActiveBatch Console and its features.
	* Implement multi-factor authentication and granular permissions to enhance security.
2. **Data Encryption**:
	* Ensure the security of your data with encryption features provided by ActiveBatch.

**Additional Features**

1. **Web Console**:
	* Access the ActiveBatch Console from any web browser using the Web Console interface. This allows you to manage jobs and workflows remotely.
2. **Integration Capabilities**:
	* ActiveBatch supports integration with a wide range of applications, databases, and platforms, enabling seamless end-to-end automation.

## Requirements For Running ActiveBatch

Running ActiveBatch requires specific hardware and software configurations to ensure optimal performance. Here's a detailed overview of the requirements and the types of servers needed to run ActiveBatch jobs:

**Hardware Requirements**

1. **Processor**:
	* Intel®, AMD®, or compatible, 2-GHz processor or higher.
	* Multi-processors are supported and utilized by ActiveBatch if available.
2. **Memory (RAM)**:
	* Minimum of 8 GB of memory.
	* Memory utilization increases with the number of plans and jobs deployed.
3. **Hard Disk Space**:
	* 700 MB required for a complete installation.
	* An additional 350 MB required temporarily during installation.
	* Database hard disk requirements start at 50 MB and will increase based on the number of ActiveBatch objects (plans, jobs, schedules, etc.).

**Software Requirements**

1. **Operating Systems**:
	* ActiveBatch supports various operating systems, including Windows, Linux, IBM AIX, z/OS, iSeries, Sun Solaris, HP-UX, HP NSK, OpenVMS, Amazon EC2, MacOS, and SCO UnixWare.
2. **Database**:
	* ActiveBatch uses "open" databases, allowing flexibility in choosing a database for use. Supported databases include SQL Server, Oracle, Microsoft Azure SQL, and AWS RDS.
3. **Additional Software**:
	* Microsoft Windows Scripting Host Version 5.6 or later.
	* Microsoft .Net Framework Version 4.8.
	* Microsoft Universal C Runtime.

**Server Configuration**

1. **Job Scheduler Machine**:
	* For environments running 15,000-30,000 job instances per day, a production environment will need:
		+ CPU: 4 CPUs (Quad-core or greater).
		+ Memory: 8 GB RAM or greater.
		+ Storage: 1-5 GB.
2. **Execution Agents**:
	* Execution agents can be installed on various operating systems to run jobs. These agents communicate with the Job Scheduler to execute tasks as per the defined workflows.
3. **High Availability and Disaster Recovery**:
	* ActiveBatch supports high availability configurations to ensure continuous operation and minimize downtime. This includes dynamic or manual no-cluster failover.

**Example Server Setup**

**Scenario**: A medium-sized enterprise running 20,000 job instances per day.

**Server Configuration**:

* **Job Scheduler Server**:
	+ CPU: 4 CPUs (Quad-core)
	+ Memory: 16 GB RAM
	+ Storage: 2 TB (to accommodate database growth and job history retention)
	+ Operating System: Windows Server 2019
* **Execution Agent Servers**:
	+ Multiple servers with similar configurations to distribute the workload and ensure efficient job execution.

By meeting these hardware and software requirements, you can ensure that ActiveBatch runs smoothly and efficiently, handling your job scheduling and automation needs effectively.

## Supported Programming Languages

ActiveBatch is highly versatile when it comes to executing scripts and integrating with various programming languages. Here are some of the key languages and scripting environments supported by ActiveBatch:

**Supported Languages**

1. **PowerShell**:
	* ActiveBatch supports PowerShell scripts, allowing you to automate tasks and manage systems using Microsoft's powerful scripting language.
2. **Python**:
	* Python scripts can be executed within ActiveBatch, making it easy to integrate with various applications and perform complex data processing tasks.
3. **Batch Scripts**:
	* Traditional batch scripts (.bat) are supported, enabling you to automate command-line tasks on Windows systems.
4. **Shell Scripts**:
	* ActiveBatch can run shell scripts (e.g., Bash, Ksh) on Unix and Linux systems, providing flexibility for cross-platform automation.
5. **Perl**:
	* Perl scripts are supported, allowing for text processing and system administration tasks.
6. **VBScript**:
	* VBScript can be used within ActiveBatch for automating tasks on Windows systems.
7. **JavaScript**:
	* JavaScript can be executed, enabling web-based automation and integration with web services.
8. **SQL**:
	* SQL scripts can be run to interact with databases, perform queries, and manage data.

## Script Optimization and Management

ActiveBatch also offers features to optimize and manage your scripts:

* **Script Vaulting**: Converts scripts into reusable objects for use in end-to-end workflows.
* **Version Control**: Maintains and protects script investments by tracking changes and revisions.
* **Low-Code API Accessibility**: Allows seamless integration with virtually any tool or technology using APIs.

## ActiveBatch Security

ActiveBatch incorporates robust security features to ensure the protection of automated processes and data. Here’s a detailed look at the security measures built into ActiveBatch and the additional security configurations users can implement:

**Built-In Security Features**

1. **Windows Security Model**:
	* ActiveBatch fully supports the Windows Security Model, including Kerberos and Active Directory. This allows organizations to use their existing accounts and groups for managing user access and permissions.
2. **Multi-Factor Authentication (MFA)**:
	* ActiveBatch supports multi-factor authentication, adding an extra layer of security by requiring users to provide two or more verification factors to gain access.
3. **LDAP Support**:
	* ActiveBatch supports Lightweight Directory Access Protocol (LDAP), enabling the use of a single, enterprise-wide directory for user authentication and administration.
4. **Granular Permissions**:
	* ActiveBatch allows granular permission settings, enabling administrators to control access to specific objects and actions within the system. Permissions can be set to allow or deny access based on user roles.
5. **Data Encryption**:
	* Sensitive data within ActiveBatch is encrypted using Enhanced Data Encryption (EDE). This includes data at rest and in transit, ensuring that sensitive information is protected from unauthorized access.
6. **High Availability and Failover**:
	* ActiveBatch supports high availability configurations with non-cluster failover, ensuring that jobs and plans are redirected to standby systems in the event of an outage or failure.
7. **Audit Trails**:
	* ActiveBatch maintains detailed audit trails for all actions performed within the system. This helps in tracking changes, identifying potential security breaches, and ensuring compliance with regulatory requirements.

**User-Implemented Security Measures**

1. **Role-Based Access Control (RBAC)**:
	* Users can implement role-based access control to define and manage user roles and permissions. This ensures that users have access only to the resources and actions necessary for their roles.
2. **Custom Security Policies**:
	* Administrators can create custom security policies to enforce specific security requirements, such as password complexity, session timeouts, and access restrictions.
3. **Integration with Cybersecurity Solutions**:
	* ActiveBatch can integrate with existing cybersecurity solutions, such as CyberArk, to enhance security measures. This includes managing privileged accounts and securing credentials.
4. **Regular Security Audits**:
	* Conducting regular security audits helps identify potential vulnerabilities and ensures that security measures are up-to-date and effective.
5. **Encryption Key Management**:
	* Administrators are responsible for managing encryption keys used for securing sensitive data. Proper key management practices ensure that data remains protected even if keys are compromised.

By leveraging these built-in security features and implementing additional security measures, organizations can ensure that their automated processes and data are well-protected within ActiveBatch.

## Circular Loop Protection

ActiveBatch has a mechanism to prevent circular loops in workflows. Circular loop protection is crucial to avoid infinite loops that can cause system performance issues or failures.

**Circular Loop Protection in ActiveBatch**

ActiveBatch includes features to detect and prevent circular dependencies within workflows. When you design a workflow, ActiveBatch checks for potential circular references and alerts you if it detects any. This ensures that workflows are logically sound and do not enter infinite loops.

**How It Works**

1. **Dependency Checks**:
	* When creating or modifying workflows, ActiveBatch automatically checks for circular dependencies. If a circular loop is detected, it prevents the workflow from being saved until the issue is resolved.
2. **Alerts and Notifications**:
	* ActiveBatch provides alerts and notifications if a circular dependency is detected. This allows users to quickly identify and correct the issue.
3. **Graphical Workflow Designer**:
	* The graphical workflow designer in ActiveBatch visually represents workflows, making it easier to spot and resolve circular dependencies. The drag-and-drop interface helps users design workflows without inadvertently creating loops.

**Example Scenario**

**Scenario**: You have a workflow that includes Job A, Job B, and Job C. Job A triggers Job B, Job B triggers Job C, and Job C triggers Job A.

**Solution**: ActiveBatch will detect this circular dependency and alert you. You can then modify the workflow to remove the circular reference, ensuring that the jobs are executed in a logical sequence without looping indefinitely.

By incorporating these features, ActiveBatch ensures that workflows are efficient and free from logical errors that could lead to circular loops.

## ActiveBatch Terminology

In ActiveBatch, the nomenclature for jobs and boxes, which are called Plans in ActiveBatch, as well as nested Plans, is crucial for organizing and managing workflows effectively. Here's a detailed explanation:

**Jobs**

**Jobs** in ActiveBatch are the fundamental units of work. They represent individual tasks or processes that need to be executed. Jobs can be of various types, including:

1. **Process Jobs**: These jobs execute a specific process or executable file.
2. **Script Jobs**: These jobs run scripts written in languages such as PowerShell, Python, or VBScript.
3. **Jobs Library**: These jobs use pre-built job steps from the ActiveBatch Jobs Library, which includes a wide range of tasks like file management, data transfer, and more.

**Job Properties**

Each job in ActiveBatch has several properties that define its behavior and execution parameters:

* **Name**: The unique identifier for the job.
* **Description**: A brief explanation of the job's purpose.
* **Schedule**: Defines when the job should be run (e.g., daily, weekly, on specific events).
* **Dependencies**: Specifies other jobs or conditions that must be met before the job can execute.
* **Triggers**: Events that initiate the job, such as file arrival, time-based schedules, or completion of other jobs.

**Plans**

**Plans** in ActiveBatch are containers that group multiple jobs together. They help in organizing and managing related jobs as a single unit. Plans can contain jobs, other Plans (nested boxes), or a combination of both. This hierarchical structure allows for complex workflows to be broken down into manageable sections.

**Nested Plans**

**Nested Boxes** are Plans within other Plans. This nesting capability allows for creating multi-level workflows where each level can represent a different stage or component of the overall process. For example, a top-level plan might represent a monthly financial processing workflow, with nested boxes for data extraction, data transformation, and report generation.

**Example of a Nested Box Structure**

1. **Top-Level Box: Monthly Financial Processing**
	* **Nested Box 1: Data Extraction**
		+ Job 1: Extract Data from Database
		+ Job 2: Extract Data from API
	* **Nested Box 2: Data Transformation**
		+ Job 1: Clean Data
		+ Job 2: Aggregate Data
	* **Nested Box 3: Report Generation**
		+ Job 1: Generate Financial Report
		+ Job 2: Distribute Report

**Benefits of Using Plans and Nested Plans**

* **Organization**: Helps in logically organizing related jobs, making it easier to manage and understand complex workflows.
* **Reusability**: Plans can be reused in different workflows, promoting consistency and reducing duplication of effort.
* **Dependency Management**: Simplifies the management of dependencies and conditions across multiple jobs.
* **Scalability**: Allows for scalable workflow design, accommodating changes and expansions without disrupting the overall structure.

By understanding and utilizing the nomenclature of jobs, plans, and nested plans in ActiveBatch, you can create efficient and well-organized workflows that streamline your automation processes.

## Using Conditionals on Jobs and Plans

Setting up conditionals on jobs and plans in ActiveBatch involves using constraints and triggers to control the execution flow based on specific conditions. Here’s a detailed explanation of how to set up these conditionals:

**Setting Up Conditionals on Jobs and Plans**

1. **Constraints**:
	* Constraints are conditions that must be met before a job or plan can execute. They are not triggers but rather prerequisites for execution.
	* **Types of Constraints**:
		+ **File Constraint**: Ensures a job or plan runs only if a specific file exists or does not exist.
		+ **Job Constraint**: Ensures a job or plan runs only if another job has completed successfully or failed.
		+ **Variable Constraint**: Ensures a job or plan runs only if a specific variable meets a defined condition.
		+ **Resource Constraint**: Ensures a job or plan runs only if a specific resource is available.
	* **Setting Constraints**:
		+ Navigate to the job or plan’s properties.
		+ Go to the "Constraints" tab.
		+ Add the desired constraints and configure their properties.
2. **Triggers**:
	* Triggers initiate the execution of jobs or plans based on specific events or conditions.
	* **Types of Triggers**:
		+ **Time-Based Triggers**: Schedule jobs or plans to run at specific times or intervals.
		+ **Event-Based Triggers**: Initiate jobs or plans based on events such as file creation, job completion, or system events.
		+ **Constraint-Based Scheduling (CBS)**: Combines constraints and triggers to control job execution based on multiple conditions.
	* **Setting Triggers**:
		+ Navigate to the job or plan’s properties.
		+ Go to the "Triggers" tab.
		+ Add the desired triggers and configure their properties.

**Example of Setting Up a Conditional Job**

**Scenario**: You want to run a data processing job only if a specific file exists and another job has completed successfully.

**Steps**:

1. **Create the Data Processing Job**:
	* In the ActiveBatch console, create a new job and name it "Data Processing Job."
2. **Set File Constraint**:
	* Go to the "Constraints" tab.
	* Add a File Constraint to check for the existence of the specific file.
3. **Set Job Constraint**:
	* Add a Job Constraint to ensure that another job (e.g., "Data Extraction Job") has completed successfully.
4. **Configure Triggers**:
	* Go to the "Triggers" tab.
	* Add an Event-Based Trigger to initiate the job when the file is created.
	* Add a Constraint-Based Scheduling (CBS) trigger to combine the file and job constraints.
5. **Save and Activate**:
	* Save the job and activate it. The job will now run only if the specified file exists and the other job has completed successfully.

By using constraints and triggers, you can create complex conditional workflows in ActiveBatch, ensuring that jobs and plans execute only when specific conditions are met.

## Advanced Features

ActiveBatch uses a variety of special features to give the user extensive flexibility when setting up jobs and plans.

1. **Job Steps Library**:
	* ActiveBatch includes a comprehensive Job Steps Library with hundreds of pre-built job steps for various systems and platforms. This library simplifies the creation of workflows by eliminating the need for custom scripting.
2. **Event-Driven Automation**:
	* ActiveBatch supports event-driven automation, allowing jobs to be triggered by various external conditions such as file events, email events, and system events. This flexibility helps streamline workflows and minimize delays.
3. **Interfaces**:
	* ActiveBatch offers versatile user interfaces, including the ActiveBatch Console, a web-based console for remote management, and a self-service portal for business users. These interfaces enhance efficiency and usability.
4. **SLA Management**:
	* ActiveBatch provides strong SLA management capabilities, allowing you to monitor and manage critical workflows from a centralized view. You can visualize key details, customize alerts, and automate remediation to ensure optimal service delivery.
5. **Reporting and Monitoring**:
	* The platform includes comprehensive monitoring, alerting, and reporting features designed to prevent job failures and delays. Customizable alerts and proactive SLA monitoring help ensure seamless end-to-end workflow execution.
6. **Security, Auditing, and Governance**:
	* ActiveBatch strengthens security around job scheduling with robust security features and real-time monitoring. It supports authorized access, granular permissions, and integrates with existing cybersecurity solutions like CyberArk.
7. **High Availability**:
	* ActiveBatch supports high availability configurations to combat outages and failures with dynamic or manual no-cluster failover.
8. **Change Management**:
	* The platform allows you to manage and synchronize objects across different environments, such as test, QA, and production, ensuring consistency and reliability.
9. **Reusable Job Templates and Variables**:
	* ActiveBatch enables the use of reusable job templates and variables, allowing you to execute the same job or plan for different workflows, which speeds up creation and editing.

**Customer Success**

ActiveBatch users benefit from a blend of resources, including self-paced training, certifications on Redwood University, and steadfast support from a dedicated team of automation experts.

**Wrap-up**

ActiveBatch is a powerful and versatile tool for workload automation and job scheduling. Its extensive features, integration capabilities, and user-friendly interfaces make it a valuable asset for organizations looking to streamline their IT operations and improve efficiency.

## Resources, Alerts, and References

Resources in ActiveBatch serve to limit the number of instances of a given job step can run at the same time. Alerts are used to alert interested parties of a job failure or some other state for a given job. References point to existing jobs and can be called like the ‘original’ job in such a way that completion triggers and constraints can be used.

**Resources** in ActiveBatch are shared objects that can be associated with multiple jobs and plans. They are used to create resource constraints, ensuring that certain conditions are met before a job or plan can be dispatched to run. Resources can be either static (constant) or dynamic (active).

* **Static Resources**: These have a constant numeric value that you assign. For example, if you have a software license that allows only five concurrent jobs, you can create a resource object named "Licensed Apps" with a value of 5. Each job that uses this software will check if a unit is available before running.
* **Dynamic Resources**: These are initialized from a data source and can change over time. For example, a dynamic resource might be linked to a SQL Server database table, and its value can be updated based on the data in the table.

**Example of a Static Resource:**

* **Resource Name**: Licensed Apps
* **Value**: 5
* **Jobs**: Job1, Job2, Job3, Job4, Job5, Job6, Job7, Job8, Job9, Job10
* **Constraint**: Each job requires 1 unit to run. If all 5 units are in use, additional jobs will wait until a unit becomes available.

**Alerts**

**Alerts** in ActiveBatch are used to notify users when specific conditions occur, such as job failures, queue disconnects, or SLA breaches. Alerts can be configured on jobs, plans, queues, and the job scheduler.

* **Alert Types**: Alerts can be configured to notify users through various methods, including email, SMS, or writing to the Alerts pane in ActiveBatch.
* **Alert Properties**: Alerts have properties such as priority (high, medium, low) and can be associated with actions that need to be taken when the alert condition occurs.

**Example of an Alert:**

* **Alert Name**: Job Failure Alert
* **Condition**: Job failure
* **Notification Method**: Email
* **Recipients**: admin@example.com
* **Priority**: High

**References**

**References** in ActiveBatch are objects that point to existing jobs or plans. They allow you to reuse existing jobs or plans in different workflows without duplicating them. This minimizes maintenance and ensures consistency.

* **Job References**: A reference to an existing job that can be used in multiple workflows. Any changes to the original job will be reflected in all references.
* **Plan References**: Similar to job references, but for plans.

**Example of a Job Reference:**

* **Original Job**: FTPCustFiles
* **Reference Job**: FTPCustFiles\_Reference
* **Usage**: The reference job can be used in different workflows, and any updates to FTPCustFiles will automatically apply to FTPCustFiles\_Reference.

## Reporting Services

ActiveBatch Reporting Services provide powerful tools for generating and managing reports related to job scheduling and execution. These services enable users to collect, process, and analyze job history data to create useful reports for monitoring and optimizing workflows.

**Key Features of ActiveBatch Reporting Services:**

**1. Instance Reporting**

* **Online Analytical Processing (OLAP)**: Allows businesses to collect and process job history (instance data) to create detailed reports. OLAP reports provide insights into job performance, execution times, and resource utilization.
* **Built-in Reports**: ActiveBatch includes built-in reports using Microsoft SQL Server Reporting Services (SSRS), Crystal Reports, and Oracle Answers. These reports offer a comprehensive view of job instances and their performance.

**2. Template Reporting**

* **Report Templates**: Users can create and customize report templates to suit their specific needs. These templates can be reused to generate consistent reports across different jobs and workflows.
* **Ad-Hoc Reporting**: The Reporting Facility allows for the creation of ad-hoc reports, enabling users to extract and retrieve information about ActiveBatch job instances on demand.

**Examples of ActiveBatch Reports:**

**Example 1: Job Execution Summary Report**

* **Purpose**: Provides a summary of job executions, including success rates, failure rates, and average execution times.
* **Components**:
	+ **Job Name**: The name of the job.
	+ **Execution Count**: The number of times the job was executed.
	+ **Success Rate**: The percentage of successful executions.
	+ **Failure Rate**: The percentage of failed executions.
	+ **Average Execution Time**: The average time taken for job execution.

**Example 2: Resource Utilization Report**

* **Purpose**: Monitors the utilization of resources, such as CPU, memory, and disk space, during job executions.
* **Components**:
	+ **Resource Name**: The name of the resource.
	+ **Job Name**: The name of the job utilizing the resource.
	+ **Usage Percentage**: The percentage of resource utilization.
	+ **Peak Usage**: The peak usage of the resource during job execution.

## Conclusion

ActiveBatch is a robust and versatile workload automation and job scheduling solution designed to streamline IT operations and enhance efficiency across various industries. With its low-code/no-code interface, ActiveBatch empowers users to automate complex workflows without extensive coding knowledge, making it accessible to both technical and non-technical users. The platform's comprehensive features include advanced job scheduling, event-driven automation, and a vast library of pre-built job steps, which simplify the creation and management of automated processes. ActiveBatch's integration capabilities are unparalleled, supporting a wide range of applications, databases, and platforms, ensuring seamless end-to-end automation. Additionally, the platform offers robust monitoring, alerting, and reporting tools, providing real-time visibility into job statuses and enabling proactive management of service level agreements (SLAs). Security is a top priority, with features such as multi-factor authentication, granular permissions, and data encryption ensuring that automated processes are protected. ActiveBatch also supports high availability configurations, ensuring continuous operation and minimizing downtime. With its user-friendly design, extensive integration options, and powerful automation capabilities, ActiveBatch is an invaluable tool for organizations looking to optimize their IT operations and drive business success.