Performance Tuning Notes

Informatica Cloud Application Integration: Process Server Running on Secure Agent
Abstract

This topic provides performance tuning notes for the Process Server running on an Informatica Cloud Secure Agent, using the Dispatch Service, server properties, agent sizing, and logging options.

Supported Versions

- Cloud Application Integration and Business Process Manager (BPM, formerly ActiveVOS)

Table of Contents

Introduction ...................................................................................................................................... 2
  Customer Variables ..................................................................................................................... 3
Server Statistics ............................................................................................................................... 3
Server Properties ............................................................................................................................. 3
  Database Connection Pool .......................................................................................................... 5
  Logging Levels and Monitoring Thresholds ................................................................................. 5
Memory ........................................................................................................................................ 6
Secure Agent Sizing ........................................................................................................................ 6
  Sizing Recommendations Summary ............................................................................................ 6
JVM Settings ................................................................................................................................ 6
Informatica Cloud Agent Engine Configuration ............................................................................ 7
Dispatch Service .............................................................................................................................. 7
  Inbound Requests ...................................................................................................................... 8
  Concurrent Execution .................................................................................................................. 8
Monitoring Dispatch Services ....................................................................................................... 8
Configuring Dispatch Services ...................................................................................................... 9
  Viewing Dispatch Service Execution .......................................................................................... 10
Managing Dispatch Services ........................................................................................................ 11

Introduction

Customers who are running processes using the Informatica Cloud Secure Agent as a private process server can set performance tuning parameters. You may need to size the Secure Agent to manage load balancing at peak transaction times.

Most tuning options are accessible in the Process Console through the Server Properties and Dispatch Service options, and several logging and monitoring levels. This article provides some background information and tips for server performance tuning using the Process Console.

For additional details on performance tuning, see:
http://www.activevos.com/developers/technical-notes

You can also find additional information on the Server Properties and Dispatch Service in the documentation here:
Customer Variables

The specific performance tuning options are highly dependent on customer variables such as:

- Type of request pattern / complexity of the process
- Whether you need persistence to enable the ability for a process to recover gracefully in case of system failure
- Volume of processing
- Number of requests to be processed simultaneously
- Available memory
- Response time of services and systems used in a process

Because of the many variables, the tuning parameters that work well for one environment may not be applicable to others.

Server Statistics

To get some insight into your baseline performance, first view the Server Statistics for the Secure Agent to help determine the best settings.

After you choose the specific Secure Agent to tune in the Console list, select Server Monitoring > Server Statistics from the Monitor menu to open up the dashboard where you can view detailed statistics:

Server Properties

The following Server Properties have particular relevance to performance:

- Work Manager Thread Pool Min/Max
- Work Manager Threads Per Process Max
- Process Count
- Resource Cache size

Note that you can make configuration changes without stopping and restarting the Secure Agent. Changes take effect immediately.
1. From the **Admin** menu, choose **Configure Server > Server Properties**.

![Configure Server > Server Properties]

2. Review and change these properties, as suggested:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Performance Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process Count</strong></td>
<td>Maximum number of processes in memory.</td>
<td>Set in proportion to the resource cache.</td>
</tr>
<tr>
<td></td>
<td>Default: 250.</td>
<td></td>
</tr>
<tr>
<td><strong>Resource Cache</strong></td>
<td>Number of WSDL files and other resources in stored cache.</td>
<td>Modifying the cache size may improve engine performance.</td>
</tr>
<tr>
<td></td>
<td>Default: 100.</td>
<td>Set this value in proportion to the process count.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>As a starting point, set the resource cache on a 2-1 ratio to the process count.</td>
</tr>
<tr>
<td><strong>Work Manager Thread Pool Max</strong> (not available if you are using an application server Work Manager with Process Server)</td>
<td>Maximum number of execution threads the engine can spawn simultaneously.</td>
<td>If the number of threads being run is equal to this value, processes can fault as no threads are available when a node needs to broadcast information to other nodes.</td>
</tr>
<tr>
<td></td>
<td>Default: 300.</td>
<td>Create a secondary pool to be used by Process Server when critical system work must be performed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you have a large number of one-way requests, try setting this to 300.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you are using this with a Dispatch Service, allow for a 2-1 ratio between the Worker Thread Pool Max and the Max Concurrent setting in the Dispatch Service.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Performance Tuning</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Work Manager Thread Pool Min</strong></td>
<td>Minimum number of execution threads the engine allocates for its work manager.</td>
<td>A simple rule of thumb is to have enough work threads to run the number of processes plus the number of simultaneous invokes that processes may execute.</td>
</tr>
<tr>
<td>(not available if you are using an application server Work Manager with Process Server)</td>
<td>Default: 25.</td>
<td></td>
</tr>
<tr>
<td><strong>Work Manager Threads Per Process Max</strong></td>
<td>Maximum number of execution threads the engine can spawn simultaneously for an individual process.</td>
<td>Unmatched Correlated Receive Timeout</td>
</tr>
<tr>
<td></td>
<td>Default: 10.</td>
<td>Use with a Dispatch Service (see below):</td>
</tr>
<tr>
<td><strong>Unmatched Correlated Receive Timeout</strong></td>
<td>Time to wait (in seconds) for a correlated message to be matched to a receive, onMessage, or onEvent activity if the message arrives before the activity becomes active.</td>
<td>As you monitor dispatch services, you can use other configuration settings to help monitor and manage an overflow of requests to the Process Server. If there appear to be issues, it could be because the Dispatch Service's Max Queued value is being exceeded.</td>
</tr>
<tr>
<td></td>
<td>If this value is exceeded, the message is discarded and a correlation violation exception is thrown.</td>
<td>If you determine that requests are being rejected based on the Max Queued value in the Dispatch Service, set this value to avoid a potentially large number of unconsumed messages on the server.</td>
</tr>
<tr>
<td></td>
<td>Default: 30 seconds.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When the timeout expires, all the data that the server has for a message is added to the server log, including the service name, operation name, and message parts.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The unmatched message is guaranteed to wait for the specified interval before it times out. The message is recovered upon server crashes and will continue to wait until it is consumed by a relevant consumer or timeout.</td>
<td></td>
</tr>
</tbody>
</table>

**Database Connection Pool**

You may also need to adjust the database connection pool settings for the server, so that the pool is at least as high as the Process Count above.

Check the database connection pool setting in your particular environment (for example, `\Tomcat\conf` or similar).

**Logging Levels and Monitoring Thresholds**

Process Console also provides several logging options, which may impact performance. For example, a higher server logging level increases contention and slows the system down. As a result, consider lowering or disabling these logging options in your production systems. You can also set higher logging levels for specific processes to decrease the contention overall.
You can also use Monitoring Thresholds to validate input and output levels and then tune for greater efficiency. However, these thresholds will slow down the system. You should therefore use them only during development and testing and disable them on production systems. You can set tracing levels on individual processes using Process Designer.

**Memory**

Finally, if you have more available memory on your server and the incoming load is high, you can consider allocating more memory. However, you can first try to increase the Process Count if the statistics indicate that active processes are waiting too long for an available slot.

**Secure Agent Sizing**

In addition to the server properties discussed above, you can also tune the Informatica Cloud Secure Agent to optimize it for specific workloads. This section provides three sizing configurations and recommendations for your JVM settings, and the agent’s process engine. JVM settings need to be configured on the agent.

**Sizing Recommendations Summary**

These are the basic sizing recommendations for the Informatica Cloud Secure Agent:

<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process Count</strong></td>
<td>50</td>
<td>150</td>
<td>300</td>
</tr>
<tr>
<td><strong>Resource Cache (MB)</strong></td>
<td>50</td>
<td>150</td>
<td>300</td>
</tr>
<tr>
<td><strong>Work Manager Min</strong></td>
<td>50</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td><strong>Work Manager Max</strong></td>
<td>250</td>
<td>500</td>
<td>750</td>
</tr>
<tr>
<td><strong>JVM Min Heap (MB)</strong></td>
<td>Default</td>
<td>768</td>
<td>1536</td>
</tr>
<tr>
<td><strong>JVM Max Heap (MB)</strong></td>
<td>Default</td>
<td>1536</td>
<td>4096</td>
</tr>
</tbody>
</table>

**JVM Settings**

The default JVM memory for Java processes is determined by the physical memory of the system. The defaults, if unspecified, are:

- Minimum heap size: 1/64 of physical memory
• Maximum heap size: the smaller of 1GB or 1/4 of physical memory
To configure your JVM, use:

```
-Xms[jvm min heap size]m -Xmx[jvm max heap size]m -XX:MaxPermSize=384m
```

where **Xms** is starting memory (Min Heap), **Xmx** is maximum memory (Max Heap), and **MaxPermSize** refers to the amount of memory set aside for the Secure Agent to load Java class files.

**Informatica Cloud Agent Engine Configuration**

To configure parameters for the Informatica Cloud agent process engine:

3. From the Informatica Cloud **Configure** menu, select **Runtime Environments** and choose the specific agent you want to configure.

4. Create an entry for each of these parameters:

   **To set** | **Enter Name as:**
   --- | ---
   Process Count: | ProcessManager/ProcessCount
   Work Manager Min: | WorkManager/DefaultWorkManager.ThreadPool.Min
   Work Manager Max: | WorkManager/DefaultWorkManager.ThreadPool.Max
   Resource Cache (MB): | ContributionManager/Catalog/cache.max

5. For each entry you create, also specify:

   **Type:** Agent Process Engine
   **Sub-type:** INFO
   **Value:** Based on the recommendations above

For example, this illustration shows the configuration for one Secure Agent:

---

**Dispatch Service**

Occasionally, a service has a heavily used process with one-way operation requests (as opposed to request-response types). One-way requests follow a fire-and-forget semantic that dispatches requests to the Secure Agent as fast as possible and does not wait for a reply. During runtime, if many thousands of these requests, plus request-response requests, try to execute simultaneously, the server resources or invoked service could be overwhelmed.

You can solve this problem by using a Request Dispatch Service. This service controls the maximum number of concurrently executing requests by putting requests in queues and dispatching them in batches. It also controls the number of new requests dispatched while allowing running processes to complete.

Process Server has a default dispatch configuration. In addition, you can create a dispatch configuration for a service or process group.

The Dispatch Service provides an enhanced degree of control and visibility over the workload for handling service requests within the process engine. For example, you can:
• Control the maximum number of concurrently executing requests for a particular service, process group, tenant, or entire engine
• Buffer overflow requests to persistent queues for reliable delivery
• View live service-level execution statistics in the Process Console
• Dynamically update the configuration to tune performance and adapt to changing load
• Implement the service without refactoring existing processes or introducing external JMS configurations

The Dispatch Service is particularly useful in situations such as:
• The process calls a back-end service where each request performs heavy-duty processing on the remote system. If too many requests are issued at once, the back-end service fails. A Dispatch Service can limit the number of process instances that are simultaneously calling the service.
• The client system sends 90,000 one-way process requests all at once to a single agent. Rather than try to process all of these requests immediately, you can store the requests and process them in volumes that can be handled by the agent. A Dispatch Service lets you throttle incoming requests and then review the heap allocation for the agent based on the load.

Inbound Requests
For each configuration created, the Dispatch Manager maintains a Dispatch Runtime that wraps and manages service-level dispatch elements for a particular service, including:
• Configuration details
• Queue Management
• Execution statistics

If you enable dispatch control, when the Process Server receives an inbound message request, the Dispatch Manager buffers the request for controlled execution through the runtime.

Note also that:
• Dispatch control applies to all inbound receives, with the exception of sub-process invokes
• HTTP, JMS, and process invokes are buffered
• One-way and request-response are buffered

Concurrent Execution
• While the maximum number of requests are executing, other requests wait in the queue
• New requests are dispatched to the engine as the processes kicked off by previous requests enter a quiescent state
• We determine that a process has gone quiescent as follows:
  - Process has been released by the process manager and is eligible to leave memory
  - Process is either in a final state, suspended, or running with no pending invoke activities running
• If a particular request is taking a long time for the process to go quiescent, the timeout value may kick in and allow another request to start nonetheless

Monitoring Dispatch Services
After you have configured a dispatch service, you can view statistics for the agent as follows:
1. Choose the agent from the list on the Console
2. Open Process Monitoring > Dispatch Service from the Monitor menu.

For each saved Dispatch Service, you can view:
• Executing: Number of requests currently executing
- **Queued:** Number of requests currently queued
- **Average Time:** Average amount of time per request for process to reach a quiescent state
- **Consumed:** Total number of requests consumed and executed
- **Rejected:** Total number of requests that were rejected by the engine
- **Status:** Current service status, which may be *Active, Suspended, or Pending Delete*

### Configuring Dispatch Services

To configure and save the policy for a new Dispatch Service for the selected agent, follow these steps:

1. From the **Monitor** menu, choose **Process Monitoring > Dispatch Service**.
2. Click **Add Configuration** to create a new profile or double-click on an existing profile to update the settings:

   ![Dispatch Configuration Dialog](image)

   - **Name**
   - **Max Concurrent**
   - **Max In Memory**
   - **Max Queued**
   - **Timeout (seconds)**
   - **Persistent**

   At runtime, the dispatch configuration used for a particular request is chosen based on matching the configuration name in the following order of precedence: Service Name, Process Group, Tenant, System Default

3. In the Dispatch Configuration dialog, set these options:
### Name

The name must match a process service name, a process group name (added within the deployment descriptor), or tenant. Tenant is applicable only with a multitenant license. Once you add and save a name, the name cannot be changed. If needed, you can delete the configuration and add a new one. If the name is not an exact process service or process group name, Process Server does not know how to use the configuration.

### Max Concurrent

Maximum number of inbound receives that can be processed concurrently. The default is 250.

When creating a dispatch service, this setting and the Max Queued setting should be the same. The only time you might want to set this value larger than Max Queued is if you define a persistent dispatch service. This kind of service stores information until there is room for it in memory.

### Max In-Memory

Maximum number of pending requests to hold in memory. The default is 2500.

### Max Queued

Maximum number of pending requests to hold in the queue (both storage and memory). The default is 2500. Any additional requests waiting to be queued will never execute.

When creating a dispatch service, this setting and the Max Concurrent setting should be the same. See Max Concurrent for more information.

### Timeout (seconds)

Maximum amount of time, in seconds, that a single request may hold up another request from being executed. The default is 300. If the timeout is exceeded, a pending request is allowed to execute.

### Persistent

If selected, buffered requests are held in storage for recovery. Otherwise, buffered requests are only held in memory.

By default, persistence is disabled. Holding requests in memory avoids additional overhead and, in any event, HTTP requests cannot leave memory.

To handle a large volume of one-way requests, enable Persistence and increase the Max Queued value.

As noted, at runtime, the dispatch configuration used for a specific request is chosen based on matching the configuration name in the following order of precedence: Service Name, Process Group, Tenant, System Default.

### Viewing Dispatch Service Execution

After you have defined a Dispatch Service or updated the System Default, you can monitor the execution and update for performance tuning based on a changing load. You can monitor live information for each configuration.

1. From the **Monitor** menu, select **Process Monitoring > Dispatch Service**.
2. Select a profile in the list.
3. For each Dispatch Service configuration, you can view:
<table>
<thead>
<tr>
<th>Executing</th>
<th>Number of requests currently running in an engine node.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queued</td>
<td>Number of requests currently queued, waiting to execute. New requests are dispatched to the engine as processes started by previous requests enter a quiescent state. A quiescent state applies to processes eligible to leave memory: final state, suspended, or running with no invoke activities running.</td>
</tr>
<tr>
<td>Average Time (ms)</td>
<td>Average execution time per request, displayed as hh:mm:ss.milliseconds. This includes the time per request for the process to reach a quiescent state.</td>
</tr>
<tr>
<td>Consumed</td>
<td>Total number of requests consumed and executed by the engine.</td>
</tr>
<tr>
<td>Rejected</td>
<td>Total number of requests that were rejected by the engine; (for example, unmatched receive, validation errors.</td>
</tr>
</tbody>
</table>
| Status | **Active**: The manager is actively dispatching queued messages to the engine.  
**Suspended Messages** remain on the queue until dispatch is resumed.  
**Pending Delete Queued** requests are waiting to be consumed. You may need to refresh the browser just after selecting Delete Configuration to remove the configuration from view. |

**Managing Dispatch Services**

You can take the following actions on each configured Dispatch Service.

1. Choose one of these options from the **Action** list and click **Execute**:
   - **Reset Statistics** resets the historical counts and averages.
   - **Suspend Execution** allows you to suspend the service.
   - **Resume Execution** allows you to resume a suspended service.
   - **Purge Queued Requests** deletes all waiting requests from the queue
   - **Delete Configuration** removes this configuration. If there are requests still queued, the status will be tagged as ‘Pending Delete’. No new requests will be added, and the configuration will be removed once all old requests have been consumed.

2. Click **Execute**.
Authors

Informatica Cloud Application Integration Team