Informatica® Cloud Application Integration
October 2024

Monitor

Informatica Cloud Application Integration Monitor
October 2024

© Copyright Informatica LLC 1993, 2024

Publication Date: 2024-10-03

Table of Contents

Preface 6

Chapter 1: The Application Integration Console 7

Chapter 2: Process Server Configuration 8

Tenant Detail 8

Properties 9

- Summary 9
- Server Properties 10
- Logging 17

Monitors 19

Receive Queue 23

Dispatch Service 24

- Dispatch Service Components 25
- Creating a Custom Dispatch Configuration 26
- Dispatch Service Configuration 27
- Monitoring a Dispatch Configuration 28
- Sample Dispatch Configurations 29
- Troubleshooting the Dispatch service 32
- Additional Controls Related to the Dispatch Service 32

Storage 32

- Scheduled Maintenance 33

System Services 34

- Alert Service 34
- Email Service 34
- Shell Service 36
- Messaging Service 36
- Datasource Service 40

Chapter 3: Deployed Assets 44

Deploying a Business Process Archive 44

Understanding Contributions 45

- Contribution Detail 47

Deployed Processes 48

Resources 49

- Viewing Catalog Resources 49

Indexed Properties 52

URN Mappings 53

Chapter 4: Process Schedules. 56
Creating a New Process Schedule. 57
Managing and Modifying Execution Schedules. 58
Process scheduler REST APIs. 59
Get Schedule API. 59
Create Schedule API. 60
Update Schedule API. 62
Delete Schedule API. 63

Chapter 5: Processes. 65
Process filters. 66
Simple process filters. 66
Advanced process filters. 69
Process View Detail Page. 77
Start Step Properties. 78
Fault Details. 78
Active Process Detail Page 80
Process Version Listing Page 89
Deployed Process Version Detail Page. 89
Understanding Process Version Life Cycles. 91
Updating a Process Version. 92
Setting a Process Version Offline or Online. 92
Setting the Logging Level. 92
Setting the Process Version Persistence Type. 93
Invoke Recovery Type. 93
Process Instance Retention. 94
Deployed Process Detail Graph. 94
Process Restart. 95
Restarting a Process from the Application Integration Console. 95
Restarting a Process from Application Integration. 96

Chapter 6: Process Server Health. 98
Performance. 98
Alerts. 99
Statistics. 99

Chapter 7: Process Metrics. 101
Alarm Queue Metrics. 101
Receive Queue Metrics. 102

Chapter 8: APIs. 104
Viewing APIs. 104

Deactivating an API. 105
Activating an API. 107

Chapter 9: Connections. 109

Starting and stopping an event source in a listener-based connection. 110

Chapter 10: Guides. 112

Guide Results Page. 113

Chapter 11: Logs. 115

Deployment Logs. 115
Server Log. 116
Setting the Process Server Logging Level. 118
Deleting the Process Server Log. 119

Preface

Read *Monitor* to learn how to monitor and manage your assets with the Application Integration Console.

CHAPTER 1

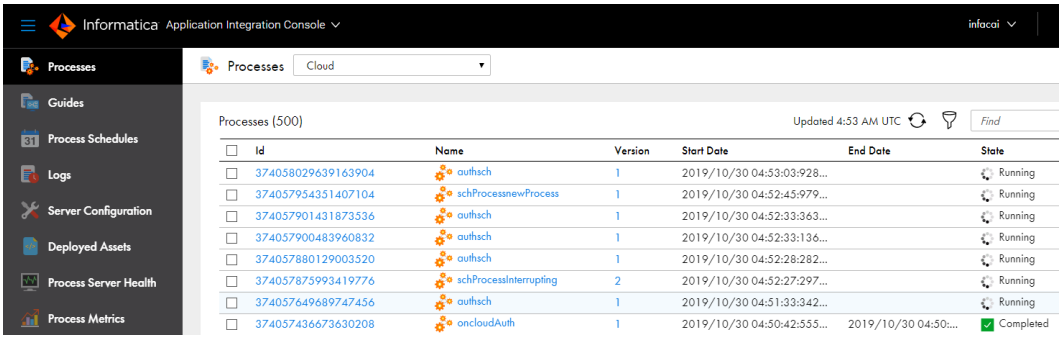
The Application Integration Console

The Application Integration Console provides options to monitor and manage your processes, whether they are running on the Informatica Cloud Server or on a Secure Agent you create in a new runtime environment.

When you use the Application Integration Console in Informatica Cloud®, the Home page looks similar to the illustration below. However, the options available depend on:

- Permissions on your user account.
- Whether you select an agent or the Cloud Server.
- Other deployment options based on the product version you are using.

Note: Because the available options vary significantly based on your situation, this online help may contain topics that are not applicable.



Note: When you monitor Application Integration assets that contain XML data, you might see changes in the sequence of nodes. This is because Application Integration maintains only the XML structure and might not maintain the same sequence of nodes.

CHAPTER 2

Process Server Configuration

View the following details on the Server Configuration page:

- Tenant or Process Server properties
- Monitor settings
- Queue settings
- Storage and scheduled maintenance settings
- System services settings

Tenant Detail

The Tenant Detail section displays information about a tenant. This page is only available for the Cloud Server.

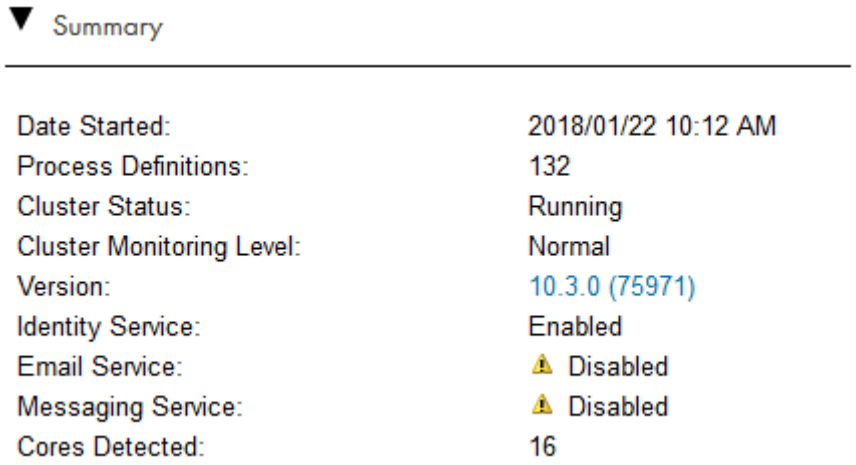
Item	Description
Tenant Context ID	The organization ID.
Tenant Name	A descriptive name.

Item	Description
Activity Execution Limit	The maximum number of Business Process Execution Language (BPEL) activities that can be executed. Use this option to suspend or terminate processes that use infinite loops and a high number of resources. The default value is 10,000. When you publish a process, Application Integration generates a BPEL, which is an executable representation of a process. You can view the BPEL representation of a process in the advanced view of a process. The activity execution limit value defines the count at the BPEL activity level and not at the process step level. A process step can get translated into more than one BPEL activity. Therefore, a process that has fewer than 10,000 steps in Process Designer can get translated into more than 10,000 BPEL activities and reach the default activity execution limit. To find the total number of BPEL activities, you can count the number of times the keyword `Executing [` appears in the process execution log. When a process reaches the maximum level, Application Integration suspends or terminates the process based on the persistence level configured for the process. If the persistence level is set to **Full** or **Persist**, Application Integration suspends the process. If the persistence level is set to **None**, **Brief**, or **Final**, Application Integration terminates the process. You must manually resume suspended processes. The activity execution count is reset to zero when you resume a suspended process. You can get around this limitation by using a Secure Agent to run the process.
Admin Groups	The names of admin groups defined for the tenant.
Service Consumer Groups	The names of service consumer groups defined for the tenant.
Console URL	The URL that is used to invoke the tenant's Application Integration Console.
SOAP Services URL	The URL to which a SOAP request can be sent.
REST Services URL	The URL to which a REST request can be sent.

Properties

Summary

The Summary tab provides an overview of the engine that executes BPEL processes.



It contains the following items.

Item	Description
Date Started	Engine start date
Process Definitions	Number of business processes (`.bpel` files) currently stored in the database
Status (or Cluster Status)	Possible statuses values for Process Server engines are Running and Stopped. Additional database messages may also display. See *Admin > Maintenance > Storage* to see more detailed information regarding the database. The status displayed reflects all engines in a Secure Agent cluster. For example, if the Secure Agent cluster has two engines, and one is stopped, the status displays `Running (1/2 running)`.
Monitoring Level (or Cluster Monitoring Level)	Level indicates a monitoring severity; that is, it indicates whether a warning or error is detected. If there is no monitoring set up or if the engine is running normally, the level is Normal. Levels include Normal, Warning, and Error.
Version	Process Server version number
Identity Service Email Service Messaging Service	A status that indicates whether a service was configured and enabled.

Server Properties

To see Process Server, select a Secure Agent.

On the Server Settings tab, you can make configuration changes without stopping and restarting the engine. When you change the properties and select Update, the changes take effect immediately. The changes are also written to the database and propagated to other engines in the Secure Agent cluster.

View and update Process Server configuration settings as shown in this topic. Note that some of these properties are the same as the Process Developer Simulation preferences.

Allow proxy user in process invocations

A process can have a policy assertion (called Run as User) that enables a named user to be a process initiator, as opposed to the normal process initiator with credentials or an anonymous user. If desired,

you can disable this property and the policy assertion is ignored. This setting applies to all new process instances and is enabled by default.

Auto create target path for Copy/To

Applies only to processes that are validated against the BPEL4WS 1.1 specification. For WS-BPEL 2.0 processes, this property can be added as an extension on a per process basis. Refer to the *Process Developer Guide* for more information.

This property determines if Process Server can create a location path for a non-existent node in a complex variable in a process instance document. When an assignment refers to a non-existent node (or to more than one node), the standard BPEL fault, `bpws:selectionFailure`, must be thrown, according to the BPEL specification.

Enabling this option allows selections to be created on-the-fly. This means an assign copy TO operation can refer to a non-existent node and assign a value to it. This option is disabled by default.

Contribution Cache

Setting this cache controls how many contributions are kept in memory. This cache avoids having to read the database too often. Generally contributions are small. However, the size depends on how many resources (that is, catalog entries) they contain and how many imports and exports they have.

Changing the default value depends on the number of deployed contributions and the amount of memory available. Default is 100 contributions. If a contribution is cached, it need not be looked up in the database. If you have more than 100, you may want to increase this value.

This cache setting doesn't require much tuning and profiling the application server is probably required to determine if changing this value would improve performance.

This cache is not set in the **Server Properties** dialog. Instead, you will need to set this in the engine configuration file, which is `aeEngineConfig.xml`, which is contained in the `activvos.war` file.

Default Work Manager Thread Pool Max

Set the maximum number of execution threads that the Process Server can simultaneously initiate for normal user work, excluding process invocations. Default is 100.

Default Work Manager Thread Pool Min

Set the minimum number of execution threads that the Process Server can simultaneously initiate for normal user work, excluding process invocations. Default is 10.

Deployment Lock Timeout (seconds)

In a clustered environment, one machine acquires a deployment lock, and the other machines must wait until deployments are complete before starting `.bpr` deployments. If you have many machines in a Secure Agent cluster, you may need to increase the timeout value because each machine deploys many system `.bprs` upon start-up. Default is 120 seconds.

Deployment Plan Cache

A deployment plan corresponds to each deployed version of a process, including associated disposition of running processes. Process versions that are active can be cached for better engine performance. Default number of plans that are cached is 100. For details regarding versions, see *Process Version Life Cycles*.

Disable bpws:selectionFailure fault

Applies only to processes that are validated against the BPEL4WS 1.1 specification. For WS-BPEL 2.0 processes, this property can be added as an extension on a per process basis. Refer to the *Process Developer Guide* for more information.

This option allows an XPath query in the FROM clause of an assignment statement to return an empty node set. If it does, the target node for the assignment is deleted.

By default, this option is not enabled, and if the query string returns an empty selection from an assign copy FROM, the process throws a `bpws:selectionFailure` fault, which is the standard response described in the BPEL4WS specification.

Invoke Work Manager Thread Pool Max

Set the maximum number of execution threads that the Process Server can simultaneously initiate for process invocations. Default is 300.

Invoke Work Manager Thread Pool Min

Set the minimum number of execution threads that the Process Server allocates for process invocations. Default is 25.

Maximum HTTP Connections/Maximum HTTP Connections Per Host

If you have several HTTP Invokes within a process, the Invokes could spawn more than 100 connections. If this occurs, you should, you should increase the default maximum number of HTTP Connections value so that multiple threads are not suspended waiting to create a HTTP Connection. (You will see service call timeouts when this happens.)

Changes you make do not occur until you restart the server.

Max Connection Timeout (seconds)

Specifies the maximum time the Process Server will wait while attempting to establish a new HTTP connection. Default is 10.

Message size limit

Sets the maximum size of the payload for a multi-part message that does not have attachments.

If you use the Informatica Cloud Server, you can set a maximum message payload size of 5 MB.

If you use a Secure Agent, the default value is 5 MB, but you can change set a higher or lower value.

Message TTL (seconds)

Specifies the amount of time between the time a message reaches the Process Server and the time when it is about to get dispatched to the target process. Default is 24 hours.

When you configure the message time to live (TTL) period and invoke a process by using the REST or SOAP endpoint, if the HTTP message is not dispatched within the message TTL period, the request fails. You see the HTTP error `503 Service Unavailable` with the following error message:

```
Message discarded having exceeded the defined TTL.
```

The message TTL is applicable to the endpoints with the following entries:

- `/active-bpel/services/`
- `/active-bpel/public/rt/`
- `/active-bpel/rt/`
- `/active-bpel/public/soap/`
- `/active-bpel/soap/`
- `/active-bpel/odata/repository/v4/OdataRepository/Execute`
- `/active-bpel/tf/`
- `/activebpel-cert/services/`
- `/activebpel-cert/rt/`
- `/activebpel-cert/soap/`
- `/activebpel-cert/odata/repository/v4/OdataRepository/Execute`

- /process-engine/rt/
- /process-engine/public/rt/
- /process-engine/services/
- /process-engine/public/soap/
- /process-engine/soap/

Message with attachments size limit

Sets the maximum size of the payload for a multi-part message that has attachments.

If you use the Informatica Cloud Server, you can set a maximum message payload size of 5 MB. The maximum size for attachments is also 5 MB.

If you use a Secure Agent, the default message payload size is 5 MB and the default attachment size is 5 MB. You can change the default sizes to set higher or lower values.

Migrate running processes even on warning

If Migrate is selected as a deployment option on a contribution, running processes are migrated to the new process version. The server checks for compatibility between the old and new process plans and generates a warning in the Process Server Log if an incompatibility exists.

Be sure that the Process Server Log is set to at least Warning Level before any migration is attempted.

If this property is disabled (the default), and if a warning condition exists for one or more running process instances, a copy of each running process instance is created, and both the pre-migration copy and the new copy are suspended. Additionally, a warning code/message is added to the Process Server Log.

If this property is enabled, no copy of the running instance is made if a warning condition exists. The only action is that a warning code/message is added to the Process Server Log.

For details, see *Migrating Running Processes when Warnings are Generated*.

Non-Persistent Process Invoke Max Timeout (seconds)

Specifies the amount of time the Process Server will wait for a request or response message to be completed. Default is 0, that is, the Process Server will wait for an unlimited amount of time.

Process Activity Execution Limit

If the process invocation fails due to an activity limit breach of 10,000, you cannot run the process for the next 15 minutes and no new instance of the process deployment will be created during this period. Default is 10,000.

You can resume the process execution after the 15-minute time limit expires or by republishing the process.

Process Count

Specifies the maximum number of processes in memory. Default is 250. Specifying 0 indicates no limit, but this is not recommended.

Process Idle Timeout

Specifies the number of seconds to wait until process state information is written to the database during idle processing times such as waiting for a reply from an invoked service. You can increase the timeout value to enhance engine performance. You can decrease the value to ensure the full process state is always in the database. Doing so avoids potential process recovery time in the event of a server failure. Default is 10 seconds.

Resource Cache

The number of WSDL files and other resources in stored cache. Default is 100. Modifying the cache size may improve engine performance. A value of -1 means unlimited caching, but this is not recommended.

Running processes suspension limit per run

Set the total number of running processes to suspend in a single scheduled run. You can set the limit to control the overload on the system work manager thread pool.

For example, the total number of running processes is 500, and you set the limit to 100. For the first scheduled run, only 100 running processes are suspended. The remaining 400 will have to wait until the next scheduled run.

Default is -1. A value of -1 means that no limit is set for suspending running processes, and all the running processes will be suspended.

Running processes suspension timer interval (minutes)

Set the time interval in minutes at which the running processes must be suspended.

Default is 1440 minutes. This means that after 24 hours, the running processes start suspending.

Screen Cache Size (MB)

Sets the size of the cache Process Server uses for storing screens used by guides. Guides can be large in size when they are published and generally not every single path in the guide is always executed. This cache holds the HTML for the most commonly used screens within a guide. This improves the time required to load a screen as the guide containing the screen need not be loaded.

Suspend long running processes after (days)

Running processes might stay in the Active Processes list indefinitely, if they are not otherwise resolved.

Enter the number of days to retain running processes, for example, 30 days. The running processes are suspended automatically after the retention period is reached.

Default is 0. This means that the running processes are never suspended. For processes running on the Cloud Server, this option is determined by Informatica.

Suspend process on invoke recovery

For invoke activities which do not complete due to the node failure, you can suspend the process upon recovery. The process is suspended at the pending invoke, and you can perform process exception management, if desired.

An individual process can override this setting with an entry in the PDD file.

Suspend process on uncaught fault

According to the WS-BPEL 2.0 specification, a process with an uncaught fault terminates.

Enable this option to suspend all processes on an uncaught fault to put them in a suspended-faulting state. You can then perform process exception management on the faulting process such as retrying or completing the faulting activity or scope.

An individual process can override this setting with an entry in the PDD file. See *Exception Management Type*.

See *Process Exception Management*.

Suspended processes termination limit per run

Set the total number of suspended processes to terminate in a single scheduled run. You can set the limit to control the overload on the system work manager thread pool.

For example, the total number of suspended processes is 500, and you set the limit to 100. For the first scheduled run, only 100 suspended processes are terminated. The remaining 400 will have to wait until the next scheduled run.

Default is -1. A value of -1 means that no limit is set for terminating suspended processes, and all the suspended processes will be terminated.

Suspended processes termination timer interval (minutes)

Set the time interval in minutes at which the suspended processes must be terminated.

Default is 1440 minutes. This means that after 24 hours, the suspended processes start terminating.

System Work Manager Thread Pool Max

Set the maximum number of execution threads that the engine can spawn simultaneously. Default is 25.

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. To be safe, you should create a secondary pool to be used by Process Server. This is done in the Application Integration Console. Process Server will only use threads in this pool when critical system work must be performed.

System Work Manager Thread Pool Min

Set the minimum number of execution threads that the engine allocates for its work manager. Default is 5. Specify enough execution threads to run the number of processes plus the number of simultaneous invokes that processes might execute.

Terminate suspended process after (days)

Suspended processes stay in the Active Processes list indefinitely, if they are not otherwise resolved.

Enter the number of days to retain suspended processes, for example, 30 days. The suspended processes are terminated automatically after the retention period is reached.

Default is 0. This means that the suspended processes are never terminated. For processes running on the Cloud Server, this option is determined by Informatica.

Unmatched Correlated Receive Timeout

Sets the number of seconds that the Process Server waits to match a correlated message to a message activity or an event activity if the message arrives before the activity becomes active. Set this value to avoid many unconsumed messages on the server.

Default is 30 seconds. If you enter an Unmatched Correlated Receive Timeout of 0 seconds, the Process Server discards all unmatched correlated messages when they arrive.

If the Process Server crashes, it recovers messages and continues to wait until a relevant consumer consumes the messages, or until a timeout occurs.

If the Process Server waits for a time that is longer than the value you specify, a timeout occurs. When a time out occurs, the Process Server adds a correlation violation error to the server log. This correlation violation error does not contain any message details. The correlation violation error contains a unique hash key called `AeSecuredLogDatakey`, a concatenation of the Engine ID, Plan ID, Queue ID, and Timestamp in milliseconds of when error occurred.

The following is a sample correlation violation error:

```
correlationViolation [AeSecuredLogDatakey=786000_123456_1000_1476434734123;tenant=ABCDEF;
planId=123456;partnerLink=MessageEvent1EventPL;operation=Initiate]
```

To view message details such as the service name, operation name, and message parts, search for `AeSecuredLogDatakey` in the `AeSecuredLogData` table.

The correlation violation error contains the hash key and not the message details because the message details might contain secure information. This two-step method secures your data.

Validate input/output messages against schema

Validates the data used in service interactions against their associated schema.

Enable this option to validate data before execution starts. Disable this option for faster execution. This option is enabled by default.

Web Service Invoke/Reply Timeout

For performance reasons, a reply activity matching a receive, as well as synchronous invokes, are timed out if they do not execute within 10 minutes. If you are receiving timeout errors, you can specify a greater amount of time to wait before a process is timed out due to a reply or synchronous invoke activity not executing within 10 minutes.

Default is 600 seconds.

Work Manager Threads For Alarms Max

Set the maximum number of threads the engine will use from the work manager to dispatch work scheduled by an alarm in a process. If there are 100's of alarms firing concurrently, all of the threads in the work manager could be used just to dispatch the alarms. If you experience performance issues or deadlocks because the alarm manager is using all of the threads, you can increase this value. Default is 5.

Work Manager Threads For Process Migration Max

Increase the thread count if you have hundreds of processes to migrate. This count is on top of execution threads for the server. Default is 50 and affects the number of threads allocated for new receives. For details, see the server property above, Migrate running processes even on warnings.

Work Manager Threads Per Process Max

Set the maximum number of execution threads the engine can spawn simultaneously for an individual process. Default is 10.

Thread pool profile for event-based connections

When you use event-based connectors such as AMQP and Kafka, you can increase the number of threads by increasing the thread pool size.

Informatica recommends that you increase the thread pool size only if you want to increase throughput because additional threads occupy some resources, and too large pools are not recommended. The thread pool size applies to all event-based connections.

You can configure the following properties to increase the thread pool size:

Thread Pool Min

Set the minimum number of execution threads allowed in the pool. Default is 10.

Thread Pool Max

Set the maximum number of execution threads allowed in the pool. Default is 20.

Queue Size

Set the maximum number of asynchronous actions that can be queued by the Process Server. Default is 1000.

When the core threads are busy, and the queue is full, Process Server adds additional threads up to the maximum thread count to handle additional tasks. Subsequently, if the queue receives additional tasks, Process Server rejects the tasks.

For example, if you push 10 tasks, they will be handled by the 10 core threads. The Process Server adds any additional tasks to the queue. When the queue reaches 1000 and you try to push more tasks,

Process Server adds up to 20 additional threads, which is the maximum number of threads that can be allocated to the tasks. After this, when the maximum pool size is reached and the queue is full, the Process Server rejects the tasks.

Note: After you change the thread pool settings, restart the Process Server for the changes to take effect.

When you configure the thread pool profile settings on both the **Server Configuration** page and the `aeEngineConfig.xml.mustache` file on the Secure Agent machine, the properties configured on the **Server Configuration** page take precedence.

JMS Exception Listener

The Process Server establishes a JMS Exception Listener for each connection it maintains. If after the server established a connection, it receives a connection failure notification, it attempts to reconnect for a specified interval and number of attempts. Default is 30 seconds (30,000 milliseconds) for 20 attempts.

Use the following SQL statements to change these values:

```
INSERT INTO AeConfigSettings (ConfigPath, ConfigValue)
VALUES ('MessagingManagers/<manager name>/ReconnectionInterval', '30000');
```

and

```
INSERT INTO AeConfigSettings (ConfigPath, ConfigValue)
VALUES ('MessagingManagers/<manager name>/ReconnectionAttempts', '20');
```

The only time you might want to change these values is when the default length of time is too short. If the Process Server cannot reconnect because the connection listeners are not notified, changing these settings won't help.

Logging

On the Logging tab, you can select logging levels for the Process Server and for processes.

Property Name	Description
Server Logging Level	By default, Process Server generates a log of server events, including server property configuration changes, BPR deployments, server stop and start, and process failures. You can configure one of the following logging levels for the Process Server: - Info (default) - Error - Warning - Critical - Verbose (includes all levels) - Off (No logging occurs)
Process Logging Level	By default, Process Server generates an execution log for running processes. You can view or download an execution log for a running or completed process. An execution log provides start and end times for activity execution and helps you troubleshoot faulted processes. You can configure one of the following logging levels for running processes: - **None:** The Process Server does not log any information. Use this option to enhance engine performance. - **Execution:** This is the default option. The Process Server logs all execution statements except for *Will Not Execute* statements. Select **Execution** to decrease the size of the log file. - **Execution with Service Data:** The Process Server logs all execution and fault information, as well as some WSIO activity information. For execution information, Process Server logs deadpath states, terminations, ready-to-execute, and so on. For WSIO, Process Server logs invokes, picks, and receives, but excludes information related to data assignment and changes. - **Execution with Data:** The Process Server logs all execution statements except for *Will Not Execute* statements, but includes variable, expression, and partner link data. Select **Execution with Data** to decrease the size of the log file. - **Full**: The Process Server logs all execution statements *Will Not Execute* statements for deadpath activities. For example, the Process Server logs all fault handling statements that are not executed. You can also set the process logging level to **System Default** on the **Deployed Process Version Detail** page. When you set the logging level to system default, the logging level for the process version corresponds to the process logging level property that is set on the corresponding Process Server. To improve processing speed, perform no logging or minimum logging. Informatica recommends that you use the **None** or **Terse** logging levels.

Property Name	Description
Max Buffer Size	The number of state changes, data changes, and other logging events that the Process Server holds in a buffer before writing them to the database. Default is **200**.
Persist Interval (seconds)	The number of seconds that the Process Server waits before flushing the buffer and writing log events from memory to the database. Default is **30**.
Min Threads	The minimum number of threads that the Process Server initiates upon startup to log process events. Default is **5**.
Max Threads	The maximum number of concurrent threads that can be active at any point of time for logging process events. Default is **150**.
Log all messages	If enabled, logs data of each WS message going to and from the Process Server, including OData requests, and SOAP envelope, headers, and body. All HTTP messages are logged to the file system.
Logging Base Directory	Root directory for log files. The default base directory is the location pointed to by the `java.io.tmpdir` property. When messages are logged, each engine will create message log files under the root directory under `<BASE_LOGGING_DIRECTORY>/engine<id>/`. The configuration for the base logging directory can contain ant-style parameters that are resolved relative to system properties and environment variables. For example, `${CATALINA_HOME}/`logs will create log files relative to the `CATALINA_HOME` environment variable used to start your Tomcat server.

Monitors

Select engine properties to monitor on the Monitors tab. For each property, you can provide a statistic and threshold that, when reached, alerts you to a warning or error condition.

You can decide the frequency and interval of monitoring periods should be, as described in the following table.

Field	Explanation
Threshold Period	Period for collecting and aggregating statistics. The default is five minutes.
Evaluation Frequency	Number of times during the threshold interval the statistics are evaluated. For example, if the threshold interval is five minutes, and the evaluation frequency is five times, then statistics are collected and aggregated once a minute during every five-minute period. The default is 5 times.
Maximum Trouble Items	Number of error/warning items per engine to display on the Monitoring page of the Application Integration Console.
Monitor Alert Service	Described after this table.

Monitor Alert Service

Add the name of the service that will run when errors and warnings occur for monitored properties. This service also runs when it automatically triggered by a MultiSite site unavailable status. (*MultiSite* require a special license.)

When errors occur, Process Server instantiates the alert service, which can then invoke an action, such as notifying an administrator that a monitored property has an error condition. The service can also monitor engine status (running or stopped).

To add a service, type in the Service name, and select **Save**.

The service name is the My Role partner link service, identified in the PDD file deployed with the BPEL process to be used as the alert service. You can find this name by looking on the page.

After you add the service, select View Details to view the BPEL process.

Selecting Server Properties to Monitor

Use the following procedure to monitor a property, press the **Add Row** button.

Set the property by filling in this row, as follows:

- Property to Monitor: Select an item from the picklist.
- Level: Select a severity: Error, Warning, or Critical-Stop Server.
- Statistic: Select a statistic. Sometimes there is only one choice for a property.
- Op: Select a relational operator for the threshold.
- Threshold: Add a non-negative integer to be used in the evaluation.

After adding properties, select **Update.**

Monitoring Properties

You can monitor the following properties:

Cluster communications issue detected (count)

The number of times an issue with broadcasting cluster messages to nodes was detected.

Critical storage exceptions (count)

Storage exceptions include:

- Problems issuing select statements
- Network communications failure
- Database communications failure
- Database user permission problems

Database connection acquisition time (ms)

Tracks the amount of wait time to get a connection from the datasource. An excessively long wait may indicate signs of trouble with the size of the connection pool. The monitoring includes maximum and average values.

The Process Server storage layer does not perform connection pooling. It relies on the storage implementation (usually a `javax.sql.DataSource`) to pool connections. Consult your application server or database documentation to address any issues with poor performance related to the connection pool size or connection acquisition time.

Deadlock retry attempts (count)

A deadlock retry can occur when the engine attempts to:

- Lock a process
- Write process state or journal entries
- Remove alarms for dispatch
- Match inbound receives
- Acquire an internal counter value such as process Id or deployment Id

Discarded unmatched correlated receives (count)

When a message with correlation properties fails to route to a running process instance and is not able to create a new process instance, the engine will keep trying to dispatch the message for the configured amount of time. There is a limit to the number of such unmatched messages that the engine will retry. This property tracks the number of messages that were discarded due to the buffer of unmatched messages being full.

The unmatched correlated receive timeout, which is shown on the Server Properties page, controls the amount of time that the engine will keep the unmatched message queuing until it is routed to a process instance. The pool of unmatched receives may fill up if this timeout is too high. However, such a problem may be an issue with process design as opposed to the timeout or buffer size.

Engine removed from cluster (count)

The number of times an engine was removed from a cluster due to it going offline, missing cluster broadcast messages.

Failed to lock process (count)

In the event of a failover, a process instance that began on one cluster node may be moved to a different node during recovery. It is unusual, but possible, that a process lock may fail, causing a process to suspend in the Suspended (programmatic) state. To address this problem, configure this property by set a Warning level for a count greater than 1 (or similar threshold).

By monitoring this property, as well as monitoring the Server Log for process recovery messages, you can determine if unwanted process suspensions are being caused by process lock failures.

Faulted/suspended(faulting) processes (count)

Number of processes that end in a faulted state or are suspended due to an uncaught fault.

You may have an expectation that processes running on this engine should not fault, and you want to be notified if they do.

Plan cache efficiency (percent)

Helps you determine if the Deployment cache setting is correct.

A deployment plan corresponds to each deployed version of a process, including associated disposition of running processes. Process versions that are active can be cached for better engine performance. The default number of plans that are cached is 100.

Plan cache removals (count)

The number of times process plans were removed from the cache.

Plan cache turnover (percent)

How often loading of new plans forced older plans out of cache.

Process cache efficiency (percent)

The engine configuration contains a count of the maximum number of processes which can be kept in memory before they are forced into storage.

A process is cached in memory if it is currently executing some logic or if it is quiescent but is being cached in anticipation of receiving another message, alarm, or response to an invoke. This property reports the percentage of processes that are read from memory versus process instances read from storage. For example, 100% indicates that all process reads are coming from the memory cache.

On the Server Properties page, you can set values for Process Count and Process Idle Timeout. The Process Count setting controls the size of the cache. The Process Idle Timeout setting controls how long to keep an idle process in the cache. If the process cache efficiency percentage is low and your processes contain bi-directional invokes or can process multiple inbound messages, you may benefit from increasing the Process Count and Process Idle Timeout. This will help keep processes in memory. However, if your processes are long-running and receive messages only periodically, a low process cache efficiency percentage is not necessarily a problem.

Process count exceeded (count)

The Server Properties include a Process Count option that specifies the maximum number of processes in memory. When the process count exceeds the value set for the Process Count, you can create an alert based on this property.

For example, if the Process Count is set to 50 and you want to be alerted when the count reaches 55, set this value to alert when the threshold is greater than or equal to 5.

Time to obtain plan (count)

The amount of time it takes to load a process plan into memory.

Time to obtain process (ms)

The time it takes to obtain a process is useful to determine if this operation is trending significantly higher under load situations. The monitoring includes maximum and average values. This property includes the time it takes to acquire a lock on a process as well as restore its state from storage if necessary.

This property works in conjunction with process cache efficiency.

Time to perform XSL transform (ms)

The time spent performing transforms within the `doXslTransform()` custom function.

Time to query Identity provider (ms)

Amount of time spent querying the identity provider in milliseconds. For example, if a request to the LDAP server to list groups for a user takes 20ms, that's the value of this metric for that instance.

This property allows you to track down slowness issues with an LDAP provider if you track a spike in time to query

Time to save process (ms)

Number of milliseconds required to save the process state and variables to the database.

A threshold can vary greatly depending on the process composition, number of variables, and size of variable data. This property only works for processes with a persistence setting of Full or Persist.

Time to validate messages (ms)

Reports the amount of time the engine spends validating input and output messages from receives, invokes and other activities.

This validation is enabled on the Configuration page labeled: "Validate Input/Output messages against schema." If enabled, all messages are validated. Validation can also be enabled or disabled on individual partner links through a policy assertion.

This property does not track the time spent in explicit variable validation through the BPEL validate activity or optional validate attribute on the assign activity.

You may wish to speed up processing by disabling all or selected message validation. If too much time is spent validating messages, you can take several steps. Start by redeploying processes with an added Message Validation policy assertion for partner links, which provides fine-grained control over specific types of messages. You can also disable message validation for all processes by disabling the Server Property.

Work manager work start delay (ms)

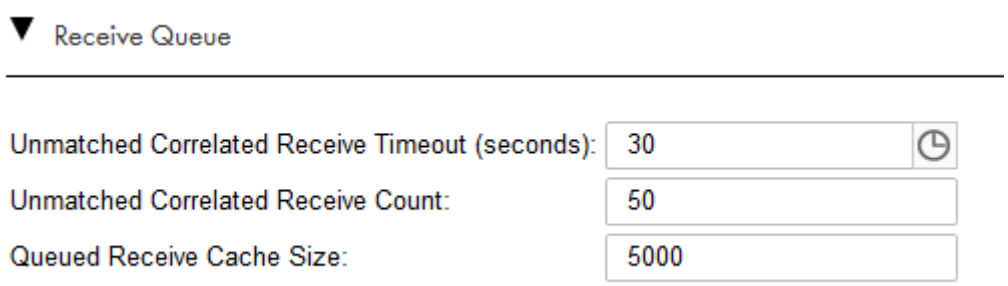
The time it takes between scheduling of a work item request and the actual start of work can help in tuning of the work manager pool.

If the time delay is trending upwards, there may not be enough threads available to handle the amount of work. The monitoring includes maximum and average values.

If you are using the default work manager, the size of the work pool can be configured on the Server Properties page. If you are using a work manager implementation provided by an application server, the size of the pool and the priorities of its threads should be configurable in your application server administration console.

Receive Queue

Use the controls on this page to set the size for how many queued receives to hold in memory to reduce the number of queries to the database. Sizing depends on server memory constraints.



The information on this page is as follows:

Unmatched Correlated Receive Timeout

The amount of time (in seconds) that the engine waits for a correlated message to be matched to a receive activity. This is only if the message arrives before the activity becomes active. If a correlated message takes longer than the specified time to be matched, the engine discards the message and you see a correlation violation exception

The default value is 30 seconds. To avoid many unconsumed messages on the server, set this value to a low number. If you set this value to specify zero seconds, the engine discards all unmatched correlated messages immediately.

When you click the **Clock icon**, you see the duration-chooser dialog box.

For more details, see *Server Properties*.

Unmatched Correlated Receive Count

The maximum number of unmatched correlated receives the engine will hold in memory at a time. When this limit is crossed, the engine rejects unmatched correlated receives; starting from the oldest, to make room for newer receives.

Queued Receive Cache Size

The size of the cache memory that holds activity information when the system is busy with other requests.

You can also change any of the values displayed for these three fields.

If events are displayed, select a receive and then select a partner link to view details. A window opens where you can see the BPEL process location in which the receive activity executes. You can also see the correlation property alias and data, if any, associated with this receive activity

Dispatch Service

Use the dispatch service to create dispatch configurations that throttle requests made to Process Server.

The requests could be HTTP/S messages, message events, or other Transport Layer Security (TLS) requests. You can queue requests, dispatch requests in batches, and control the maximum number of requests that run concurrently.

For example, you have a frequently used process that quickly dispatches asynchronous requests to Process Server. At runtime, if thousands of these requests execute simultaneously, the result can overwhelm the server and the service being invoked. Use the dispatch service to send requests in batches and to ensure that only a few requests are processed at the same time.

You can use the dispatch service for services that run on the Secure Agent. You cannot use the dispatch service for services that run on the cloud server.

Create a dispatch configuration for a service or process group. At runtime, Application Integration chooses the dispatch configuration in the following order of preference:

1. Service Name
2. Process Group
3. System Default

For example, the service `processA_event1` has the following dispatch configurations that apply to it:

- `processA_event1` (Service Name)
- `GroupABC` (Process Group)
- `SystemDefault`

When Process Server executes `processA_event1`, the `processA_event1` dispatch configuration applies.

The dispatch service applies to all requests except for subprocess invokes that contain boundary events. For example, if a subprocess step contains a message event, the dispatch service will not apply to the subprocess.

However, if you use a service call step to add a subprocess, the dispatch applies to all subprocess configurations, including subprocesses with boundary events.

Default and Custom Dispatch Services

Process Server has default dispatch services.

The following table shows the default dispatch services and their purpose:

Default Dispatch Service	Purpose
SystemDefault	Applies to all services.
avBusinessConnectionRESTService	Applies to service connectors.
avCreateAnyEntityService	Enables the host runtime to PUT records using a database connection.
avDeleteAnyEntityService	Enables the host runtime to DELETE records using a database connection.
avHostEnvironmentRuntimeAccess	Enables the host runtime to access records using a database connection.
avHostRuntimeCall	Enables the host runtime to GET records using a database connection.
avProxyToProcess	Enables a process to call other processes as a services.
jmsEnqueueService	Applies to the JMS automated step.
sfHostEnvironmentRuntimeService	Enables the host runtime to access a Salesforce database using a Salesforce connection.

To create a custom dispatch configuration, go to **Server Configuration** and select a Secure Agent. Then, go to **Queues** > **Dispatch Queue**.

Dispatch Service Components

The dispatch service uses the concurrent pool and the in-memory and persistent queues to control the flow of requests to Process Server.

The dispatch service uses the following pool and queues:

Concurrent Pool

The concurrent pool stores requests that Process Server processes at the same time. The number of listeners available to process requests depends on the Max Concurrent value.

In-Memory Queue

The in-memory queue stores requests in the dispatch service memory. The dispatch service moves requests into the in-memory queue when the Max Concurrent value has been reached.

When the concurrent pool has space, requests move from the in-memory queue to the concurrent pool.

You loose requests in the in-memory queue if the Secure Agent restarts unless you enable persistence.

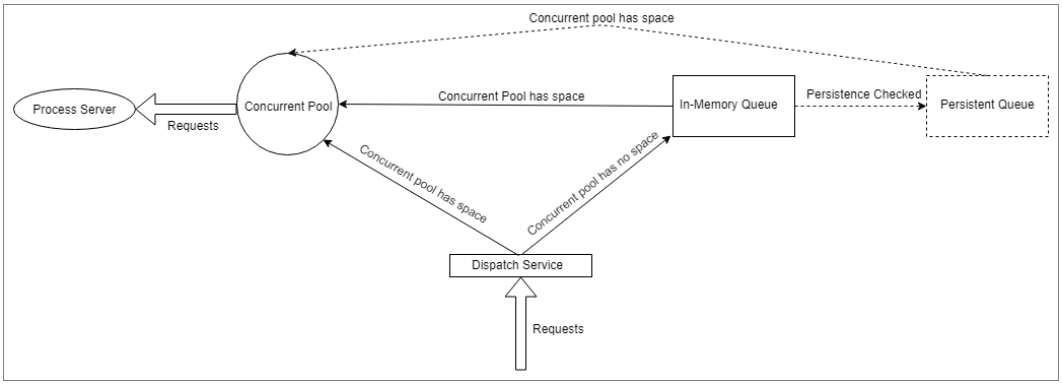
Persistent Queue

The persistent queue stores in-memory queue requests if you enable persistence. The persistent queue holds requests in the database.

Requests move from the persistent queue to the concurrent pool when a listener frees up.

If the Secure Agent restarts and requests are only in the in-memory queue and not in the persistent queue, the requests are lost.

The following image shows how the dispatch service throttles requests:



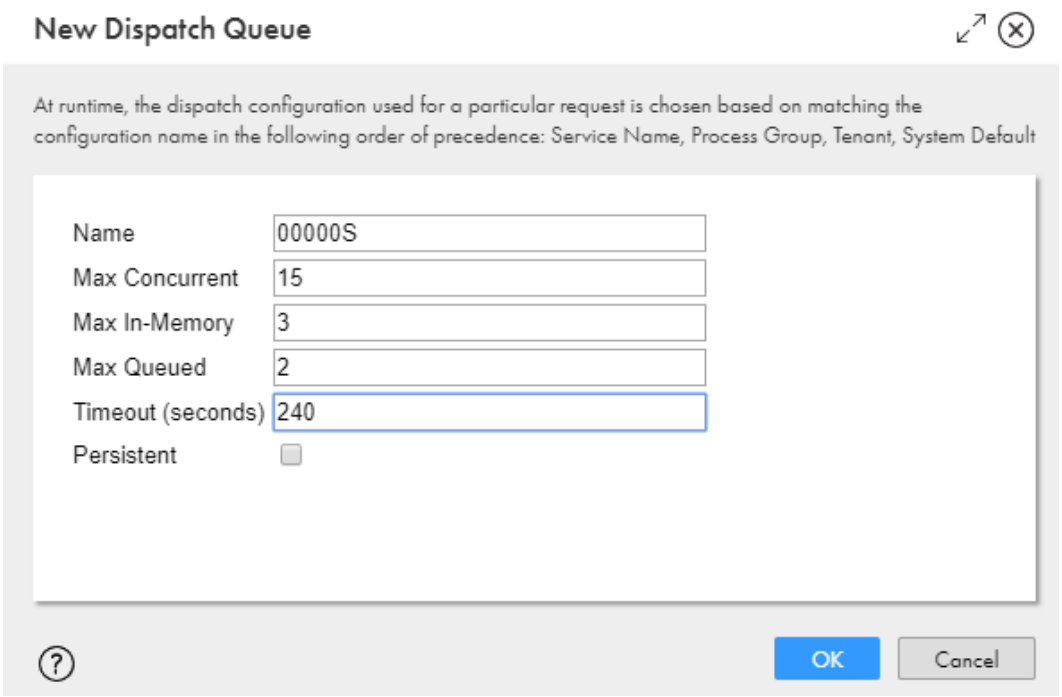
For more information, see the Sample Dispatch Configurations on page 29.

Creating a Custom Dispatch Configuration

Perform the following steps to create a custom dispatch configuration:

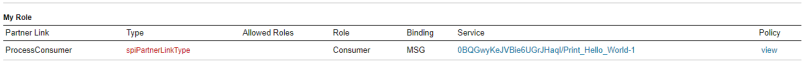
1. Log in to Application Integration Console.
2. Go to **Server Configuration** and select a Secure Agent.
3. Go to **Queues** > **Dispatch Queues** > **Add**.
4. In the **New Dispatch Queue** dialog box, configure properties and then click **OK**.

The following image shows a sample **New Dispatch Queue** dialog box for the 00000S service:



Dispatch Service Configuration

Configure the following properties to create a custom dispatch configuration:

Property	Description
Name	The name of a service definition. To view a service definition name, go to **Processes** > **Select Process Version Number** > **My Role** > **Service column**. For example, in the following **My Role** section of the **Deployed Process Version Details** page, the service name is `0BQGwyKeJVBie6UGrJHaqI/Print_Hello_World-1`:  My Role Partner Link | Type | Allowed Roles | Role | Binding | Service | Policy ProcessConsumer | spiPartnerLinkType | Consumer | MSG | 0BQGwyKeJVBie6UGrJHaqI/Print_Hello_World-1 | view If you enter a service definition name, the dispatch configuration governs all instances of the service, regardless of the process that contains the service. You cannot change the name of a saved dispatch configuration. To edit the name, delete the dispatch configuration and then create a new one.
Max Concurrent	The maximum number of requests that Process Server can hold in the concurrent pool. Default: 15.
Max In-Memory	The maximum number of requests that Process Server can hold in the in-memory queue. If you enable persistence, the dispatch service rejects requests that exceed the **Max Queued** value. If you do not enable persistence, the dispatch service rejects requests that exceed the **Max In-Memory** value. Default: 300.
Max Queued	The maximum number of requests in the persistent queue. If you enable persistence, the dispatch queue service rejects requests that exceed the **Max Queued** value. If you do not enable persistence, the dispatch service rejects requests that exceed the **Max In-Memory** value. Default: 300.
Timeout (seconds)	The maximum amount of time, in seconds, that a single request can take to execute within the dispatch service. If a request takes longer that the timeout value to execute, it moves out of the dispatch service and the next pending request executes. If a request times out of the dispatch service, Process Server continues to execute outside of the dispatch configuration. As a result, you might see different values on the **Processes** list and the **Server Configuration** > **Queues** > **Dispatch Queue** list. Default: 300 seconds.
Persistent	Select the **Persistent** option to save pending requests to the persistent queue. If you select this option and the Secure Agent restarts, the dispatch configuration continues to execute queued requests. If you do not select the **Persistent** option, pending requests are saved only in the in-memory queue and are lost if the Secure Agent restarts. Default: Not selected.

For information about how you can use these properties to throttle requests, see the Sample Dispatch Configurations on page 29.

Monitoring a Dispatch Configuration

Use the **Dispatch Queue** page to monitor dispatch configurations

The following table describes the columns that on the **Dispatch Queue** page:

Column	Description
Name	The dispatch configuration name.
Executing	The number of requests running under the dispatch configuration.
Queued	The number of requests queued by the dispatch configuration. Requests move from **Queued** to **Executing** when any of the following events occur: - A request completes. - A request times out. - A request moves out of the queue because you purged queued requests.
Average Time (ms)	The average time, in milliseconds, for which a listener in the concurrent pool works on a request.
Consumed	The number of requests that have passed through the dispatch configuration. This number does not include rejected messages.
Rejected	The number of requests that were rejected by the dispatch configuration. This number does not include timed out requests. For details, see Timeout (Seconds)on page 27.
Status	The status of the dispatch configuration. A dispatch configuration can be in the following states: - **Active:** The dispatch configuration is running and can send queued requests to Process Server. - **Suspended:** The dispatch configuration is paused. Pending queued requests remain in the queue until dispatch configuration is resumed. - **Pending Delete:** There are queued requests that need to be consumed. If you see this status, delete the dispatch configuration and refresh the browser.

The following table describes the actions that you can take on one a dispatch configuration:

Action	Description
Reset Statistics	Sets the **Executing** and **Average Time (ms)** values to zero.
Suspend Execution	Stops sending requests to Process Server. Requests remain in a queue until you resume the dispatch configuration.
Resume Execution	Resumes dispatching new requests to the engine.
Purge Queued Requests	Deletes all pending requests from the in-memory queue and the persistent queue. Process Server does not execute these requests. Use this action in unusual circumstances that require clearing the queue.
Delete Configuration	Removes the dispatch configuration. The status will be **Pending Delete** until you refresh the browser. Requests do not go to Process Server after you select **Delete Configuration**. If there are queued requests, the configuration is removed after all requests are consumed.

Sample Dispatch Configurations

This section contains an explanation of some sample dispatch configurations.

Example 1

The following table shows a sample dispatch configuration:

Property	Value
Max Concurrent	15
Max In-Memory	200
Max Queued	1000
Timeout (seconds)	55

The dispatch configuration receives 10,000 requests.

The following table describes the events that occur when each request takes 50 seconds or 60 seconds, and if persistence is enabled or not enabled:

Persistence/ Time Taken per request	50	60
Not Enabled	- Every 50 seconds, 15 requests move from the **Queued** column to the **Executing** Column. - The dispatch service processes 200 requests and rejects 9800 requests. Process Server does not execute the rejected messages. - All 200 requests are successfully processed because the **Timeout** value is higher than the time taken to complete a request - The value in the consumed column is 200 and the value in the rejected column is 9800. - If the Secure Agent restarts before 200 requests are processed, queued requests are lost because they were saved to the in-memory queue and not to the persistent queue.	- Every 55 seconds, 15 requests move from the **Queued** column to the **Executing** Column. This is because the **Timeout** value is lower than the time that a request takes to complete. - The dispatch service processes 200 requests and rejects 9800 requests. Process Server does not execute the rejected messages. - The 200 requests do not complete within the purview of the dispatch service because the **Timeout** value is lower than the time taken to complete a request. Process Server continues to execute the timed-out messages after the messages leave the dispatch service. - The value in the consumed column is 200 and the value in the rejected column is 9800. - If the Secure Agent restarts before 200 requests are processed, queued requests are lost.
Enabled	- Every 50 seconds, 15 requests move from the **Queued** column to the **Executing** Column. - The dispatch service processes 1000 requests and rejects 9000 requests. Process Server does not execute the rejected requests. - The dispatch service processes 1000 requests successfully because the **Timeout** value is higher than the time taken to complete a request. - The value in the consumed column is 1000 and the value in the rejected column is 9000. - If the Secure Agent restarts before 1000 requests are processed, pending requests execute after the restart. This is because requests were saved to the persistent queue.	- Every 55 seconds, 15 requests move from the **Queued** column to the **Executing** Column. This is because the **Timeout** value is lower than the time that a request takes to complete. - The dispatch service processes 1000 requests and rejects 9000 requests. Process Server does not execute the rejected requests. - The 1000 requests do not complete within the purview of the dispatch service because the **Timeout** value is lower than the time taken to complete a request. Process Server continues to execute the timed-out messages after they leave the dispatch service. - The value in the consumed column is 1000 and the value in the rejected column is 9000. - The value in the consumed column is 200 and the value in the rejected column is 9800. - If the Secure Agent restarts before 1000 requests are processed, pending requests execute after the restart. This is because requests were saved to the persistent queue.

Example 2

The following table shows a sample dispatch configuration:

Property	Value
Max Concurrent	4
Max In-Memory	20

Property	Value
Max Queued	10
Timeout (seconds)	7

The dispatch configuration receives 50 requests.

The following table describes the events that occur when each request takes 50 seconds or 60 seconds, and if persistence is enabled or not enabled:

Persistence/ Time Taken per request	5	10
Not Enabled	- Every 5 seconds, 4 requests move from the **Queued** column to the **Executing** Column. - The dispatch service processes 20 requests and rejects 30 requests. Process Server does not execute the rejected requests. - The dispatch service successfully processes 20 requests because the **Timeout** value is higher than the time taken to complete a request - The value in the consumed column is 20 and the value in the rejected column is 30. - If the Secure Agent restarts before 20 requests are processed, queued requests are lost because they were saved to the in-memory queue and not to the persistent queue.	- Every 7 seconds, 4 requests move from the **Queued** column to the **Executing** Column. - The dispatch service processes 20 requests and rejects 30 requests. Process Server does not execute the rejected requests. - The 20 requests do not complete within the purview of the dispatch service because the **Timeout** value is lower than the time taken to complete a request. Process Server continues to execute the timed-out messages after they leave the dispatch service. - The value in the consumed column is 20 and the value in the rejected column is 30. - If the Secure Agent restarts before 20 requests are processed, queued requests are lost because they were saved to the in-memory queue and not to the persistent queue.
Enabled	- Every 5 seconds, 4 requests move from the **Queued** column to the **Executing** Column. - The dispatch service processes 20 requests and rejects 30 requests. Process Server does not execute the rejected requests. - The dispatch service successfully processes 20 requests because the **Timeout** value is higher than the time taken to complete a request - The value in the consumed column is 20 and the value in the rejected column is 30. - If the Secure Agent restarts before 30 requests are processed, pending requests execute after the restart. This is because requests were saved to the persistent queue..	- Every 7 seconds, 4 requests move from the **Queued** column to the **Executing** Column. - The dispatch service processes 20 requests and rejects 30 requests. Process Server does not execute the rejected requests. - The 20 requests do not complete within the purview of the dispatch service because the **Timeout** value is lower than the time taken to complete a request. Process Server continues to execute the timed-out requests after the requests leave the dispatch service. - The value in the consumed column is 20 and the value in the rejected column is 30. - If the Secure Agent restarts before 30 requests are processed, pending requests execute after the restart. This is because requests were saved to the persistent queue.

Troubleshooting the Dispatch service

When you use the dispatch service, you might face situations where you are unable to move forward.

The number of requests in the Executing column of the Request Dispatch Service page is different from the number of processes running on the Active Processes page. I'm sure that my requests have not timed out.

Potential Reason: You have invoked a process that calls a subprocess.

Potential Solution: On the **Deployed Process Version Detail** page, change the **Persistence Type** setting for the process and subprocess and retry the process invoke.

Toggle between persistent (**Full** or **Persist**) and non-persistent (**Final**, **Brief**, or **None**) settings

I invoked a process 5 times but I see hundreds of requests in the Queued column. These requests are moving slowly and I think my Secure Agent is going to crash.

Potential Reason: You have invoked a process that calls a subprocess that in turn calls a subprocess. This layering might go deep.

Potential Solution: Create multiple dispatch services for the different sub processes that your process calls. Each dispatch configuration runs and throttles services individually, which speeds up service execution.

Additional Controls Related to the Dispatch Service

As you monitor dispatch services, you can use other configuration settings to help monitor and manage an overflow of requests to Process Server. If there appear to be issues, it could be because the dispatch service's Max Queued value is being exceeded.

Use the following commands to determine if requests are being rejected:

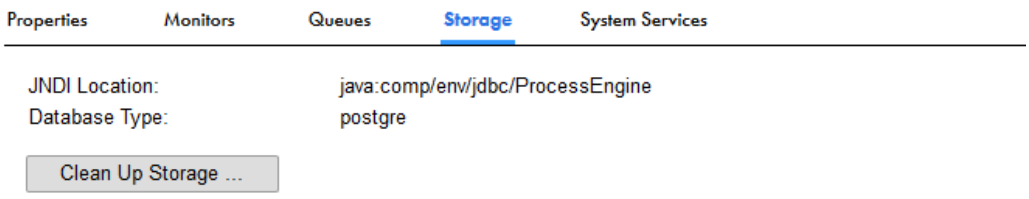
- On the **Server Configuration** > **Properties** > **Server Settings** tab, you can set a value for Unmatched Correlated Receive Timeout. For details, see *Server Properties*.
- On the **Server Configuration** > **Monitor** tab, you can add a property for Discarded unmatched correlated receives. For details, see *Monitors*.
- Check the Process Server log for errors related to rejected requests. For details, see *Server Log*.

Storage

Note: You only see this command if you have selected an agent.

Process Server includes persistent storage based on the database settings you configured during installation. You must configure a database before running the engine.

The Storage page displays database configuration properties and allows you to maintain the database.



The displayed properties are:

Property	Explanation
JNDI Location	The Java Naming and Directory Interface (JNDI) context that specifies where to look for the database. For example,`jdbc/ActiveVOS`
Database Type	The type such as `mysql`. For a list of supported types, see the "Prerequisites" topic in *Process Server Install, Configure, and Deploy* in this help.
User Name	Username, if required, for Process Server's access to the database

Scheduled Maintenance

Use the **Storage** tab to specify when and how often to delete items from the database.

Delete Schedule	
Frequency	- **None** resets the engine for no scheduled maintenance - **Intraday**, **Daily**, **Weekly**, or **Monthly** sets the frequency schedule
Every	- For **Intraday**, select minutes or hours and the number. For example, select 90 minutes. - For **Weekly**, select one or more days of the week. For example, select Monday to delete database items every Monday. Select Monday and Thursday to delete every Monday and Thursday. - For **Monthly**, select the first, second, third, fourth, or last day of the month. For example, select Last Monday to delete items on the last Monday of the month.
Start Time (GMT)	Type in a time of day, or select the Date Chooser to enter a time. In the Date Chooser, select Now to enter the current time.
Delete Offline Contributions	Select this setting to delete old process versions.
Delete Completed Processes **Delete Faulted Processes**	Select these settings to delete old processes that completed normally (Completed) or abnormally (Faulted).
Default Faulted Process Retention	Enter the days, hours, or minutes for which Process Server retains faulted processes in the database. The default is 72 hours (3 days). **Note:** You can specify a **Process Instance Retention** value on the **Deployed Process Version Detail** page. This value is specific to a process and overrides the general **Default Faulted Process Retention** value. For information about the **Process Instance Retention** value, see the *Deployed Process Version Detail* topic in the *Managing the Catalog and Resources* section of *Monitor*. If you configure an email service to receive email alerts for faulted processes, you receive a maximum of one email per hour for each faulted process. For example, you have two processes that fault every five seconds, process A and process B. You will receive two emails per hour, one for process A and one for process B.
Delete Server Log Entries Older Than	Enter the days, hours, or minutes for which Process Server retains Server Log data. For example, if you select a **Server Log Retention** period of 30 days, Process Server retains Server Log data for the last 30 days and deletes earlier Server Log data.

After you set up a maintenance schedule, select **Update** Save to save your settings.

System Services

Alert Service

You can configure an email or a custom service to run when a process stops because of a fault.

The Alert Service is available for processes that run on a Secure Agent or the Cloud Server.

To receive alerts, set the **Suspend processes on uncaught fault** property described in *Server Properties* or enable this option for processes defined in Process Designer.

Alert services are specific to the Cloud Server or the Secure Agent you select on the Application Integration Console. When you use Informatica Cloud, you can configure one Alert Service for the Cloud Server, and one for each Secure Agent to receive alerts.

You can choose one of two options when you configure an Alert Service:

- **Send Email**. Use this option to send an email to a single specified recipient when the suspended or faulting process state occurs. You must configure an Email Service before you use this option.
- **Custom Service**. Use this option to deploy a custom process. For example, you can deploy a process that notifies an administrator of a process fault. You can create a custom service in the following ways:
  - If you use the Process Developer, specify the service name that appears in the PDD file used with the custom service in the **Service** field. This service name appears on the **Service Definitions** page of the Application Integration Console. For details on how to define a custom alert service, see the *Process Developer* help.
  - If you use the Process Designer, create a process and do the following steps:
    1. Set the **Start Event** to **Event**.
    2. Next to **Event Source Name**, select `System Events`, and then **Runtime Alert Service**.

    Note: With the Process Designer, you can define more than one process with a start event of Runtime Alert Service. However, you configure one process with the Alert Service.

Select **Enable Alert Service** to use the Alert Service and click **Update** to save your changes.

Email Service

You can configure an email service to use, for example, in connection with an alert service triggered when processes are suspended on an uncaught fault. The email service settings enable you to send an email to a specified administrator.

If you execute processes designed with Process Designer, you can use Application Integration Console to configure one email service to use for processes that run on a Secure Agent and another email service for your organization's processes that run on the Cloud Server. You can then use the Send Email option when you define an Alert Service to send notifications when a process is suspended on an uncaught fault.

To define an email service, configure the following properties:

Property	Description
Enable SMTP Service	Select to enable the email service after you have configured all the properties.
Host	Enter the email server's DNS name such as mail.mydomain.com or an IP address such as 192.168.1.1.

Property	Description
Port	Enter the port to use for communication between the Process Server and the email server. The default value is 25.
Authentication Type	Select password-based authentication or OAuth authentication.

To create the email connection using password-based authentication, configure the following properties:

Property	Description
Username	Required. Enter the name used to log in to your email server, usually the account name. This field is optional depending on whether the SMTP server requires credentials. For example: notifyme@mydomain.com
Password	Required. Enter and confirm the password for the user name.
Security	(Optional). Select a security protocol, if desired: - TLS (Transport Layer Security) - SSL (Secure Sockets Layer) Ensure that you configure the port to enable the appropriate security transport.

To create the email connection using OAuth authentication, configure the following properties on the connection creation page:

Property	Description
Authorization URL	Required. Enter the OAuth authorization URL for the email service that is used to authorize the user request. For example: `https://login.microsoftonline.com/xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx/oauth2/v2.0/authorize`
Token Request URL	Required. Enter the OAuth token request URL that handles token requests. For example: `https://login.microsoftonline.com/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/oauth2/v2.0/token` The refresh token expires in 90 days. The user must authenticate again and publish the connection before the token expires.
Client ID	Required. Specify the identifier value from the OAuth provider.
Client Secret	Required. Enter the client secret to connect to the email application.
Scope	Required. Specify the scope. The scope in OAuth authentication limits an application's access to a user's account. You can select multiple scopes for a single client. To enter multiple scopes, separate each value with a space. For a Microsoft Outlook email account, enter the following scope: `https://outlook.office.com/SMTP.Send offline_access`
Authorization Status	Indicates the current status, the name of the user who authorized, and the last time when the authorization was completed.
Authorize Access	Click to initiate the authorization workflow using OAuth.

Test the configuration, and then save it.

Shell Service

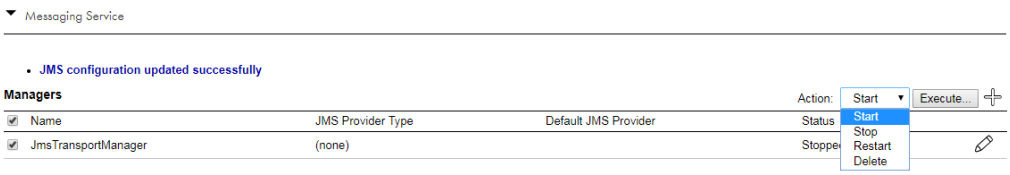
The Shell Service option enables you to define processes that include a shell command in a Service Call step, for processes that run on a Secure Agent.

After you enable the shell service, create a process with a Service step and add the shell command as a system service.

Messaging Service

You only see this tab if you have selected a Secure Agent.

Use this page to configure parameters for one or more JMS Messaging Services so that Process Server can establish connections to the JMS provider.



Use the Action list to control the messaging service's execution. First, select the Manager Name in the list and choose the Action (Start, Stop, Restart, Delete). Then, click **Execute**.

Setting Up a Messaging Service

To set up a messaging service, click **Add** and enter the following properties:

Default JMS Provider

For the provider selected in JMS Provider Type, enable this provider as the default, if desired. You can select one provider as the default, if you are setting up multiple services.

JMS Provider Type

List of configuration templates that contain some pre-defined default settings for the type of JMS provider being used. Theoretically, any provider that provides JNDI access to JMS resources can be used. If the specific provider is not on the list, you can select **Other JMS** to populate the configuration with some commonly used, generic JNDI properties.

Connection Factory Name

The JNDI name of the JMS connection factory.

Connection User

Include user credentials when creating connections on the connection factory. With providers hosted by an application server, this is usually not required since authentication takes place when accessing objects through the JNDI context. MQ Series is one provider that requires connection credential.

Connection Password

Password for the above user.

Send Empty Credentials

For MQ Series, when connecting to a remote queue manager when authentication is not enabled. You need to send empty strings ("") as username/password on the connection or it will fail with a security exception.

Maximum Total Connections

Each JMS Manager maintains an internal pool of connections to enhance performance when interacting with a remote JMS provider.

Sets the maximum number of active connections allowed at any one time. This includes connections used for asynchronous listeners configured under Queues and Listeners and Topics and Listeners as well as those used for invoke activities. If the maximum is reached, clients must wait until a connection is returned to the pool.

Setting the maximum value to -1 indicates that the manager may create as many connections as needed, with no upper limit so clients never have to wait. Change this value from the default of -1 if more connections are being created than the JMS provider can handle.

This value should be set to one of the following:

- -1 (unlimited)
- A number higher than the total number of listeners for both queues and topics and plus some connections for invokes

Maximum Free Connections

Sets the number of unused connections the manager retains at any one time. If a connection is returned to the pool, and there are already the maximum free connections sitting idle, the connection is closed. This allows the total number of connections from the pool to shrink and grow as necessary. Setting the maximum free connections to 0 prevents the Process Server from holding onto connections. Each client receives a newly created connection.

Ensure that the free connections value is lower than the value for Maximum Total Connections.

Delivery Mode

This setting controls whether or not the JMS provider persists messages to storage for all processes. An individual process can have a different persistence setting, which overrides this setting.

Enable this setting to persist messages in the event of a JMS failure. When this mode is enabled, Process Server instructs the JMS provider to ensure that a message is not lost in transit in case of a JMS provider failure. It is logged to stable storage.

Note that persistent delivery requires that your JMS provider be configured with storage. Also, there is usually a performance hit with persisting messages.

The default is disabled, which means the messages are sent with non-persistent delivery mode. This mode does not require any knowledge of how a provider is configured for storage.

Time to Live (ms)

Specifies the amount of time for which an unconsumed message remains on a queue. If a message will become obsolete after a certain period, you might want to set the time to live period. The expiration of obsolete messages conserves storage and computing resources.

Default is 0, which means that the default for the provider is used. Typically, this means that messages never expire and remain on the queue forever.

If you configure the time to live period and invoke a process by using the REST endpoint. If the HTTP message is not consumed within the time to live period, the request fails. You see the HTTP error `503 Service Unavailable` with the following error message:

```
Message discarded having exceeded the defined TTL.
```

Priority (int)

Specifies a non-negative integer for a message handling priority. Default is 0, which means that the default for the provider is used. An individual process can specify a priority, which overrides this setting.

JMS defines a 10-level priority value with zero as the lowest and nine as the highest. Clients should consider 0-4 as gradients of normal priority and 5-9 as gradients of expedited priority. Priority is set to four, by default.

Initial Context Properties

The set of name-value pairs used to establish a connection to the server hosting the JMS resources for access via JNDI lookup.

As an example, by selecting "BEA Weblogic" as the provider type, a set of default initial context properties that are generally used by WebLogic clients is displayed. Update the values for the URL, username, and password to match your environment.

Queues and Listeners

To enable connectivity with external clients and services over JMS, configure connections to an external JMS server.

- **Queue Name**/ **Topic Name** (*required*). Descriptive name for the configuration
- **JNDI Location** (*required*). Location name used for JNDI lookups
- **Listener Class** (*required*). Message listener class name. The listener class is responsible for dispatching messages to Process Server.
  If you are deploying processes created with *Process Developer*, use the listener class name:

  `com.activee.rt.mom.jms.transport.AeJmsBpelListener`

  If you are deploying processes created with *Process Designer*, use the listener class name:

  `com.activevos.rt.socrates.web.mom.jms.AeRealTimeJmsListener`

  If needed, you can create a custom listener class that extends the default listener class to include custom behavior. Specify the custom listener here.
- **Listener Count** (*required*). Number of connections to keep open. When the server starts, the JMS manager creates instances of this class that serve as asynchronous consumers on the destination. Each asynchronous consumer has its own connection to the JMS server. The number of consumers and connections created is controlled by the listener count.
- **Selector** (*optional*). JMS message selector string.
- **Default Service** (*optional*). Specifies the name of the BPEL service to use when the target service cannot be determined from the addressing headers or message properties.
  Note: One alternative to specifying a service is to include the JMS message property `JmsTargetService = "myRoleServiceName"` in the incoming message. Another alternative is to include a service name as a query parameter in a wsa:To header in a myRole partner link (for example, `<destination JNDI name>?<servicename>)`. These scenarios are for receiving XML messages over JMS.
- **Run-As Identity** (optional). Specifies an identity for a JMS request. The identity is a role-based or group membership. If not specified, the request runs anonymously. Using this property is analogous to specifying an identity for a message-driven bean in `ejb-jar.xml`. Use this option if you want the JMS listener to invoke processes that have been secured with the Allowed Roles restriction in the Process Deployment Descriptor (pdd). Another use case is using the aeRunAs header for Human Task (B4P) operations. This case allows you to specify a RunAs Identity with the abTrust security role.
- **Tenant Context** Applicable for a MultiTenant-licensed server.
- **XA Transaction** (optional). If enabled for a supported platform, and if required setup has been performed, indicates that a JMS read from the queue and the process engine should participate in the same transaction. For a discussion, see "Enabling XA Transaction Handling" on page 39.

- **Rollback on Error**: Select **Rollback on Error** if you want to decline messages that encounter an exception, or failed messages, and move to the next message. For example, the exception could be because of an authentication failure.
  If you select **Rollback on Error**, you must configure how the message server handles failed messages. For example, you can configure the message server to redeliver failed messages 10 times. Or, you can configure the server to move failed messages to a dead letter queue and decide what to do with them at a later point.

  For example, add the following policy entry to configure Apache ActiveMQ server to move failed messages to a dead letter queue:

  ```
  <policyEntry queue=">">
      <deadLetterStrategy>
      <individualDeadLetterStrategy queuePrefix="DLQ."
  useQueueForQueueMessages="true" processNonPersistent="true"/>
     </deadLetterStrategy>
  </policyEntry>
  ```

  Here, if the original queue is `userIDs`, the dead letter queue that Apache ActiveMQ server creates is `DLQ.userIDs`.

  The **Rollback on Error** option does not apply to messages that were delivered to target processes. A message is considered successfully processed if it invokes a target process, even if the process fails.

  For new queues and listeners, the **Rollback on Error** option is selected by default.

Topics and Listeners

This section is where the listeners that receive messages on behalf of the Process Server are configured. At a minimum, a single listener bound to a JNDI location on the server is needed to dispatch incoming messages to the server.

If more destinations are required to service requests, new definitions can be added by selecting **Add Queue** or **Add Topic**.

Websphere Users

If you select IBM WebSphere as a provider, and Process Server is running on WebSphere, the Queue and Topic configuration tables are not shown.

Configuration of Queue and Topic listeners is not available with the WebSphere JMS provider. WebSphere JMS explicitly forbids creating asynchronous consumers outside of a message-driven bean deployment when running on a WebSphere server.

Process Server provides a sample application for a message-driven bean deployment.

Enabling XA Transaction Handling

When you are using JMS transport for inbound requests, the only way to absolutely guarantee an *exactly once* message delivery is to use a Distributed (XA) transaction surrounding commits of process state to the Process Server database and the read of the JMS message from the queue. Without an XA transaction, it is possible, although extremely rare, to receive a duplicate request if the server goes down at the precise moment when the inbound receive was committed to the Process Server database, but the JMS transaction has not been committed. In this case, the message remains on the queue and may be retried, resulting in a duplicate process request.

You can enable Distributed (XA) Transaction handling for a JMS Messaging Service if your processes require it. Enabling this feature requires several manual steps, outlined below in Required Setup.

Note: XA transactions can be relatively expensive, so you may see an adverse affect on performance if you choose to enable them.

Using distributed transactions, Process Server can retrieve a message from a message queue and update the database in a single transactional unit adhering to the ACID (Atomicity, Consistency, Isolation and Durability) criteria. This feature guarantees that messages processed from JMS are processed once and only once.

Supported Platforms

XA Transaction processing is supported on all Process Server application servers that include a JTA transaction manager. Te only one that does not is Apache Tomcat.

All Process Server databases (MySQL, DB2, Oracle, and SQLServer) are supported.

Note: Some databases do not enable XA support by default, and this support needs to be enabled by a database administrator. Consult your database documentation to determine if and how XA transactions are enabled.

Required Setup

Follow these guidelines to set up XA transactions:

1. Ensure that your database is configured to support XA transactions.
2. Create a separate JDBC datasource that supports XA transactions. Consult the documentation for your database and application server for datasource configuration details. By default, Process Server looks for a datasource whose JNDI name is `jdbc/ActiveVOSXA`.
   Note: Process Server uses a separate datasource configuration for XA transactions since mixing XA and non-XA transactions within the same connection pool can often cause problems.
3. Set up an XA-aware JMS connection factory that enlists with the JTA transaction manager hosted by the application server. Consult the documentation for your JMS provider for details. Specify this connection factory on the JMS Messaging Service page of the Application Integration Console.
4. Check the **XA Transaction** box in the Messaging Service window that appears when you create a new messaging service.

Datasource Service

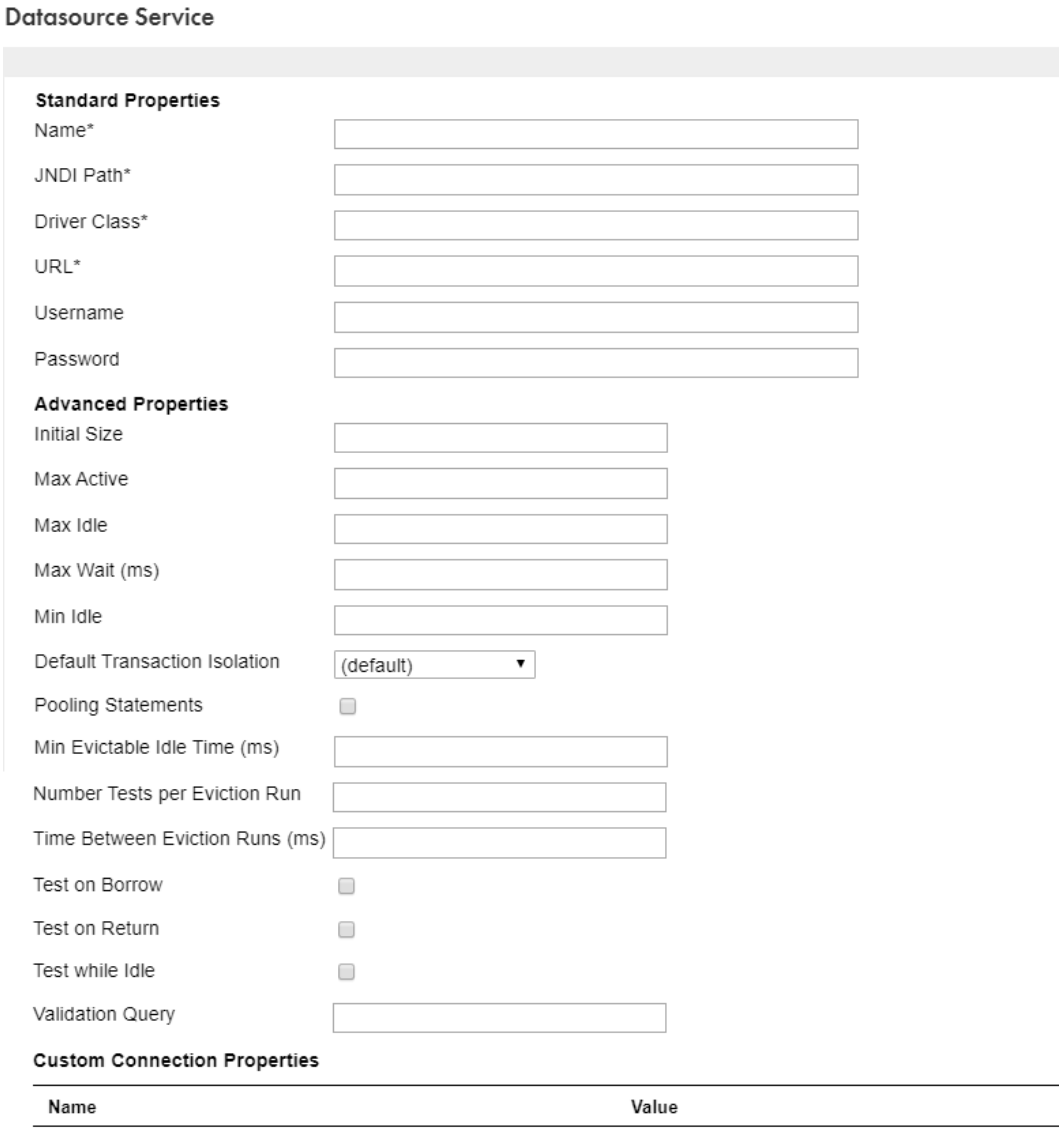
You only see this service if you have selected a Secure Agent.

Many processes interact with relational databases. The Datasource Service lets you create a "pool" of open connections between all of the application's current users. The number of users actually performing a request at any given time is usually a very small percentage of the total number of active users. Also, only when the request is being processed is there a need for a database connection. Here, you create a service that logs into the DBMS, and handles user account issues.

Note: The Datasource service is based upon the Apache Commons DBCP. See http://commons.apache.org/proper/commons-dbcp/index.html for more information.

To add a new datasource service, click **New**.

Enter the following properties to configure a datasource service:



After you make changes, you click **Test Connection** to make sure that the connection is defined and configured correctly.

Standard Properties

The standard properties are:

Name

The name of the service you are defining or editing

Driver Class

The fully qualified Java class name of the JDBC driver to be used.

JNDI Path

The JNDI name to where this Datasource is bound.

Password

The connection password passed to the JDBC driver when establishing a connection.

URL

The connection URL passed to the JDBC driver when establishing a connection.

Username

The connection username passed to the JDBC driver when establishing a connection.

Advanced Properties

The advanced properties are:

Default Transaction Isolation

The default state of connections created by this pool, which is one of the following:

- NONE
- READ_COMMITTED
- READ_UNCOMMITTED
- REPEATABLE_READ
- SERIALIZABLE

Initial Size

The initial number of connections that are created when the pool is started.

Max Active

The maximum number of active connections that can be allocated from this pool at the same time. If this value is negative, there is no limit.

Max Idle

The maximum number of connections that can remain idle in the pool without extra ones being released,. If this value is negative, there is no limit

Max Wait (ms)

The maximum number of milliseconds that the pool waits for a connection if there are none available. A value of -1 means to wait indefinitely.

Min Evictable idle Time (ms)

The minimum amount of time an object may sit idle in the pool before it is eligible for eviction by the idle object evictor (if an evictor exists.

Min Idle

The minimum number of connections that can remain idle in the pool without extra ones being created. A value of zero means that none will be created.

Number Tests per Eviction Run

The number of objects to examine during each run of the idle object evictor thread (if it exists).

Pooling Statements

Enable prepared statement pooling for this pool

Test on Borrow

When checked, objects are validated before being borrowed from the pool. If the object cannot be validated, it is dropped from the pool, and an attempt is made to borrow another.

Test on Return

When checked, objects are validated before being returned to the pool.

Test While Idle

When checked, objects are validated by the idle object evictor (if one exists). If the object cannot be validated, it is dropped from the pool.

Time Between Eviction Runs (ms)

The number of milliseconds to sleep between runs of the idle object evictor thread. If this value is not positive, no idle object evictor thread runs.

Validation Query

The SQL query that will validate connections from this pool before returning them to the caller. If specified, this query must be an SQL SELECT statement that returns at least one row.

Custom Connection Properties

Use this section to add other properties that connection requires. This information is sent to the JDBC driver when the service is creating a connections. Click the **Add** button to add a row into which you enter the property's name and value.

CHAPTER 3

Deployed Assets

Use the **Deployed Assets** page to view and manage contributions, deployed processes, catalog resources, indexed properties, and URN mappings.

Deploying a Business Process Archive

The Deploy dialog allows you to add a business process archive (.bpr file) to the Cloud Server or a Secure Agent using Application Integration Console. The .bpr file, created in Process Developer, is similar to a web archive and contains all the resources needed to deploy your project resources so they are available at runtime.

You deploy one .bpr archive at a time, but the .bpr can include all the BPEL files, deployment descriptors, partner definition files, WSDL definitions, and other resources that you need.

Note: You can't deploy the Business Process Archive (.bpr) file on the Cloud Server if the .bpr file contains the following extensions: .mp4, .mkv, .exe, .php, .jsp, .pdf, .mp3, .bpr, .zip, .rar, .tar, .py, .java, .cpp, .bash

To Deploy the .bpr file:

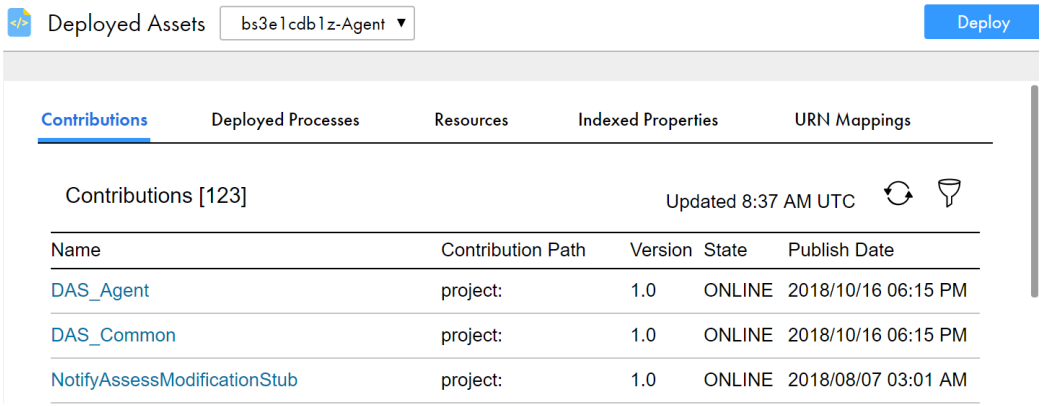
1. Log in to Informatica Intelligent Cloud Services℠ using an account that has administrative privileges.
2. Select Application Integration Console.
3. Select Deployed Assets.
4. In the upper-right corner of the page, click **Deploy**.
5. Click **Browse** to select the `.bpr` file.
6. Choose the Cloud Server or your Secure Agent from the **Deploy Target** list.
7. Click **Deploy**.

The server validates the files contained in the .bpr and stores the files in the database. The Logs page shows any errors, warnings, and information about the deployed process files.

After you deploy a .bpr file, you can view details for deployment descriptors, partner definition files, BPEL files, indexed properties, WSDL definitions, and schema files by accessing the specific resource in the Deployed Assets page.

Understanding Contributions

The Contributions tab lists all business process archive (BPR) deployments. Their names are based upon the Process Developer project from which they were deployed.



Just as Process Developer keeps together all files in an orchestration project, a contribution is a unit of the deployable processes and resources from a project. It is a unique deployment to the server catalog. As developers update their projects, they can deploy new contributions while maintaining earlier ones on the server.

Here is some information about contributions:

- Each contribution is named (by default) as `project:/project name` as a Process Developer location hint.
- A contribution can contain multiple processes, which can be managed together or separately.
- Each new version of a contribution supersedes an earlier version and becomes the current (online) version. However, you can rollback an online contribution to an earlier version. This contribution tracking system allows for easy management of processes and their resources.
- There is only one current (online) contribution. All contributions can become the online version.
- The current (online) contribution can be rolled back to an earlier version.
- Process versions are associated with a specific version of WSDL, schema, and other resources. Every BPEL process is associated with the set of resources it is deployed with.
- If you delete a contribution, all associated files and the deployment log are deleted, giving you an easy way to clean out the database during testing
- Multiple developers can deploy the same process and resources without colliding with one another.

For details, see *Contribution Detail*.

What Happens When a Contribution is Updated

When a BPR contribution is deployed for the first time, Process Server tags the contribution and every file in the deployment as version 1.0, and the contribution is online. Subsequently, as developers make changes, they may update, add, or delete files and then repdeploy the same BPR. The new version becomes the online version and the old version is offline pending, eventually becoming offline when all executing processes complete.

The following scenarios illustrate what happens when a contribution is updated.

Contribution changes from V1 to V2 -- Process1 is modified, Process2 is removed

Deployment 1 project:/MyProject V1	Deployment 2 project:/MyProject V2	Changes on the Server
`process1.bpel`	`process1.bpel` (modified)	V2 is online, V1 is offline or offline pending, if it has associated process instances.
`process.private.wsdl` `process.public.wsdl`	not modified	V1.01 is online. V1 is offline. With each deployment, WSDL and XSD are incremented to avoid cache issues.
`process2.bpel`	(not included)	The process is no longer in the catalog.

Contribution changes from V1 to V2 -- All original files are removed, a new one is deployed

Deployment 1 project:/MyProject	Deployment 2 project:/MyProject V2	Changes on the Server
(`myAVOS.avcconfig` does not exist)	`myAVOS.avcconfig`	This file is deployed as an "Additional Resource" and is the only file deployed in the same BPR as previous deployment. V1 is online. This is the only file online because it is the only file in Contribution V2. To get all files back online, redeploy the BPR and include all files.
`process1.bpel`	(not deployed)	BPEL file is removed
`process.private.wsdl` `process.public.wsdl`	(not deployed)	WSDL files are removed

For each contribution, basic version information is displayed, as shown in the table.

Item	Description
Contribution	The base URI, usually the Process Developer project name, such as `project:/myProject`
Version	Deployed version
State	By default, online state is displayed; for information, see the description of all states below this table
Date	Date the contribution was deployed
Group	Group this contribution belongs to, if any, as added in the Export Business Process Wizard in Process Developer
Deployer	Name of person (or system) who deployed, if authentication is required for deployers. The default is anonymous.

Contribution Versions

Contribution versions can have one of the following states:

- Online—By default, the newest contribution (highest numbered) is the online (current) version.

- Offline Pending—A version is offline pending in the following cases: if you set the version offline and there are running process instances; a newer version is deployed while there are running process instances; or the contribution's exported namespaces are referenced by other contributions.
- Offline—When all process instances of an offline pending version complete, a contribution is offline.

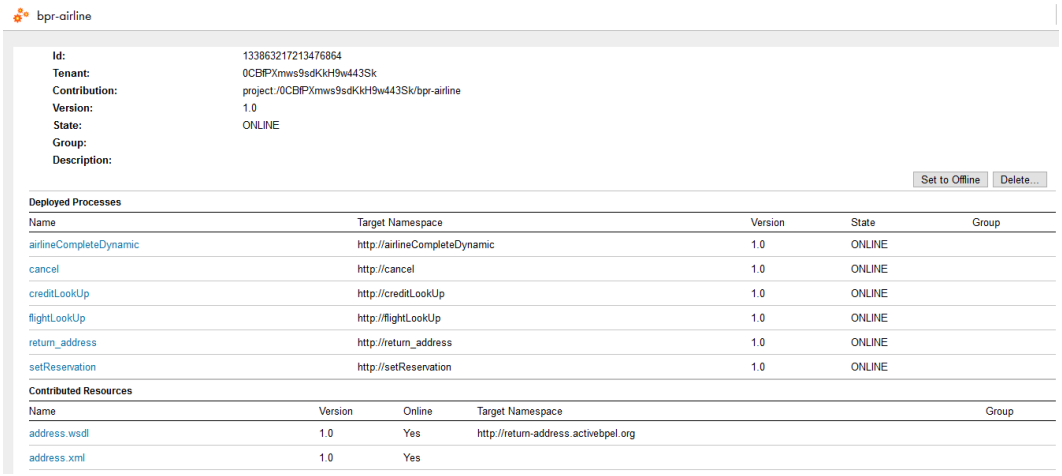
Selection Filters

You can select the following Selection Filters to view a subset of contributions:

- State—One of the contribution versions states just listed.
- Deployed between—Enter or select the starting and ending deployment dates.
- Name—Type in the contribution name. You can use wildcards, such as tutor*.*, and select **Submit**.
- Group—If the contribution was deployed with a group name, this filter is available. Type in a group name, and select **Submit**. There is a default System group, which is hidden by default. Remove the checkmark from Hide System and select **Submit** to display System contributions.

Contribution Detail

The Contribution Detail page displays a selected contribution version. You can change the state of the contribution or delete the contribution.


bpr-airline

Id: 133863217213476864
Tenant: 0CBfPXmws9sdKkH9w443Sk
Contribution: project:/0CBfPXmws9sdKkH9w443Sk/bpr-airline
Version: 1.0
State: ONLINE
Group:
Description:

Set to Offline | Delete...

Deployed Processes

Name	Target Namespace	Version	State	Group
airlineCompleteDynamic	http://airlineCompleteDynamic	1.0	ONLINE	
cancel	http://cancel	1.0	ONLINE	
creditLookUp	http://creditLookUp	1.0	ONLINE	
flightLookUp	http://flightLookUp	1.0	ONLINE	
return_address	http://return_address	1.0	ONLINE	
setReservation	http://setReservation	1.0	ONLINE	

Contributed Resources

Name	Version	Online	Target Namespace	Group
address.wsdl	1.0	Yes	http://return-address.activebpel.org	
address.xml	1.0	Yes		

Setting the State to Offline or Online

If the contribution state is online, it is the current version, and you can select **Set to Offline**. If there are other contribution versions, the next higher or lower version is set to online. If process instances are running against the BPEL files in the contribution, the contribution is set to Offline Pending. When all process instances complete, the state changes to Offline.

If the contribution state is offline, select **Set to Online**. The contribution's files become the current version. If any BPEL was deployed with an online (effective) date, the process state is online pending.

Note that the state of BPEL processes within a contribution is the same as the state of the contribution itself, but you can individually select a BPEL process from an online contribution and change its state. For example, if three processes are included in an online contribution, you can take one, two, or all three processes offline.

Deleting a Contribution

Select **Delete** to remove the entire unit of files from the catalog. If running processes exist, the contribution goes to the offline pending state until the processes complete.

When you delete a contribution, the following events occur:

- All files in the contribution are deleted.
- The deployment log is deleted.
- All completed active processes attached to the contribution are deleted.

Contribution Detail Information

The information within this page is grouped into sections, and these are described in the following list:

- **Contribution Detail**
  Each contribution is assigned a deployment Id. The name, state, version, and group details are the same as shown on the Contributions list page. A Description is available if it was provided during the BPR export in Process Developer.
- **Deployed Processes**
  List of BPEL files deployed, if any, and their current state. .
- **Contributed Resources**
  List of files deployed in the contribution. Examples of these files are WSDL, schema, HTML, XSL, XML, CSS, images, i18N properties files, Process Central files, event documents, Java jars, and report definitions. The version is the updated version associated with this contribution.

  WSDL and schema are always updated when they are deployed, even if identical to an earlier version. Other resource only have their version numbers incremented if their content has changed.

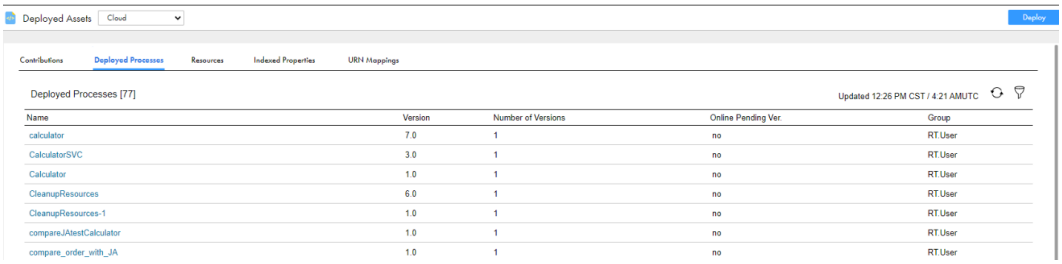
  When you click on one of these links, Process Server displays its source, if possible. For example, it may display XML or HTML.
- **Exported Namespaces**
  Namespaces exposed by a resource in the contribution for use by other contributions.
- **Imported Namespaces**
  Namespaces from other contributions that this contribution is dependent on, if any. Note that if this contribution depends on another contribution, delete this contribution first.
- **Deployment Log**
  Link to the deployment log for this contribution.

Deployed Processes

The **Deployed Processes** tab shows a list of processes that have been deployed to the cloud or to a Secure Agent.

Note: Unlike the **Processes** page, where you see an entry for each process instance, you see an entry for each process created on the **Deployed Processes** tab.

The following image shows a sample **Deployed Processes** tab:



When you select a deployed process name, you see the Process Version Detail page. For more information, see Process Version Detailon page 89.

Resources

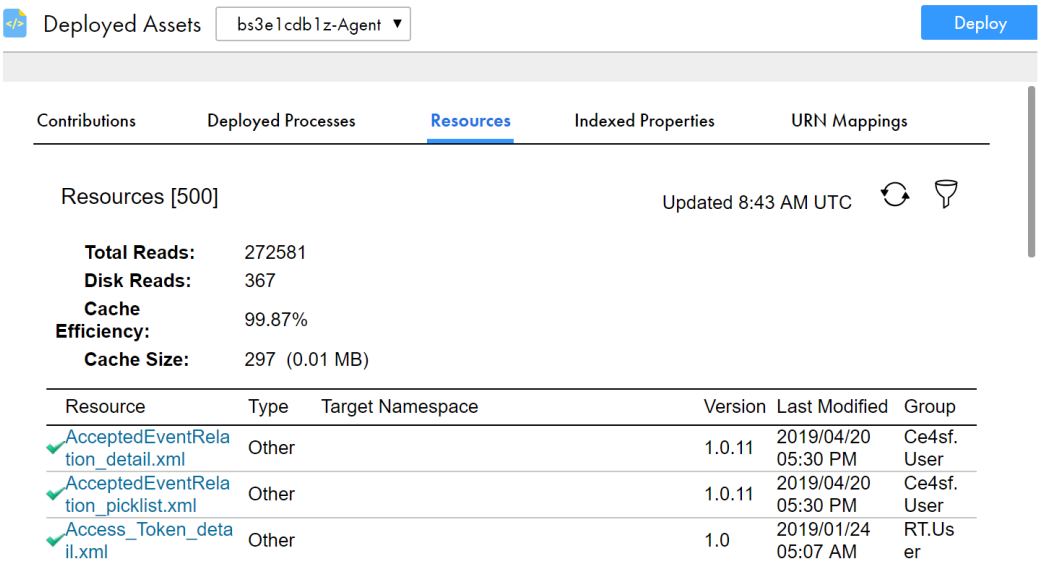
The **Resources** tab contains the following resource types:

Resource Type	Description
Central Configs (on-premises only)	Configuration files governing the display of folders and other filters for Process Central Tasks, Forms, and Reports.
Function Contexts	POJO custom functions imported into processes.
HTML Documents	Forms for initiating processes as well as task forms for People activities. Used in Process Central.
Images	Image source files referenced in HTML pages.
Java Jars (on-premises only)	Interfaces for partner services.
Report Definitions (on-premises only)	System and user-defined reports.
Schema Documents	XSDs imported into processes.
WSDL Documents	WSDLs imported into processes.
XQuery Modules	XQuery custom functions imported into processes.
XSL Documents	Transformational style sheets for custom functions and the older Inbox application renderings.
Others	- XML files for custom functions, sample data, numerous other configuration files - .properties files for multilingual support - JavaScript files for forms

Viewing Catalog Resources

Any resource can be accessed by any deployed BPEL process and only one copy is maintained.

The following image shows the Resources tab:



When you view the **Resources** tab for a Secure Agent, it displays the following details on how the node is accessing the catalog (the details do not display if you view the Resources for the Cloud Server):

Item	Description
Total Reads	The number of reads to retrieve resource information during process execution (in cache or not).
Disk Reads	The number of reads made to resource files not in the cache expressed as an absolute number and percentage of Total Reads.
Cache Efficiency	The percentage of resources that are read from memory versus read from storage. For example, 100% indicates that all resource reads are coming from the memory cache.
Cache Size	The number of resource files in stored cache. The default is 100. You can set cache size on the **Admin** > **Configure Server** > **Server Properties** . Modifying the cache size may improve engine performance.

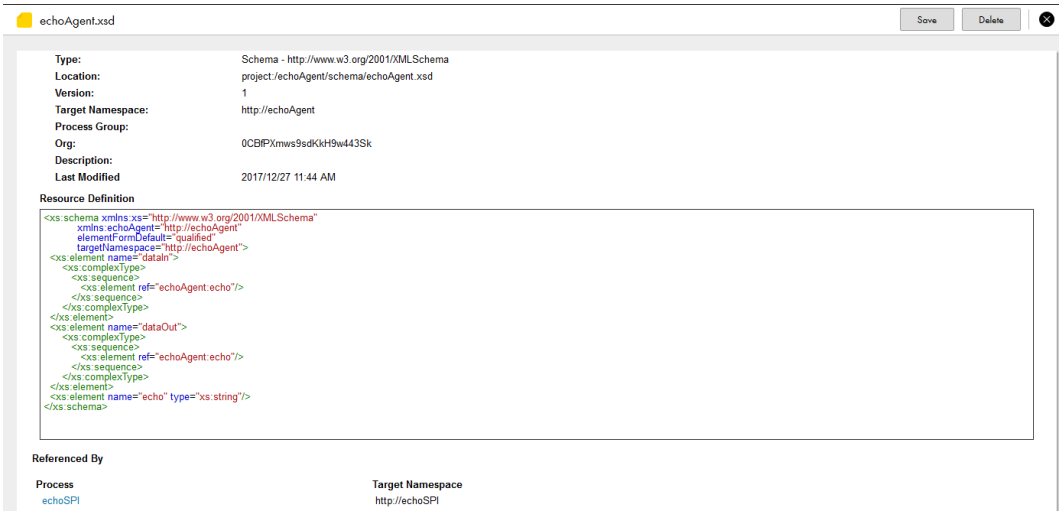
The Resources list shows the type, name, and namespace for the resource. Rest your mouse on the Resource name to view the physical location where the resource was loaded from.

The information about a resource on this page is the same as that is displayed in the Resource Detail page. Clicking on a resource link displays the Resource Detail page.

Resource Detail Page

The **Resource Detail** page shows the same information that is on the Resources page for each resource. It also displays the XML source code.

The following image shows a sample Resource Detail page:



Property	Description
Name	The name of the resource
Class	The class from which the resource was instantiated
Location	The physical location from where the resource is loaded. This helps to uniquely define the location when the deployment descriptor was created and can be used to have multiple resource files of the same name deployed to the engine. The WSDL location is referenced in the `.pdd` file.
Version	The version of the resource. This is either automatically created by the Process Server or assigned to in the `.pdd` file.
Target Namespace	Target namespace in the resource
Process Group	Group name that was added in the `.pdd` file; this is used to organize resources
Description	Description that was added in the `.pdd` file
Last Modified	The date at which this resource was last changed

The **Resource Detail** page shows the process versions referencing this resource for some resources.

Selection Filters

Use Selection Filters as follows:

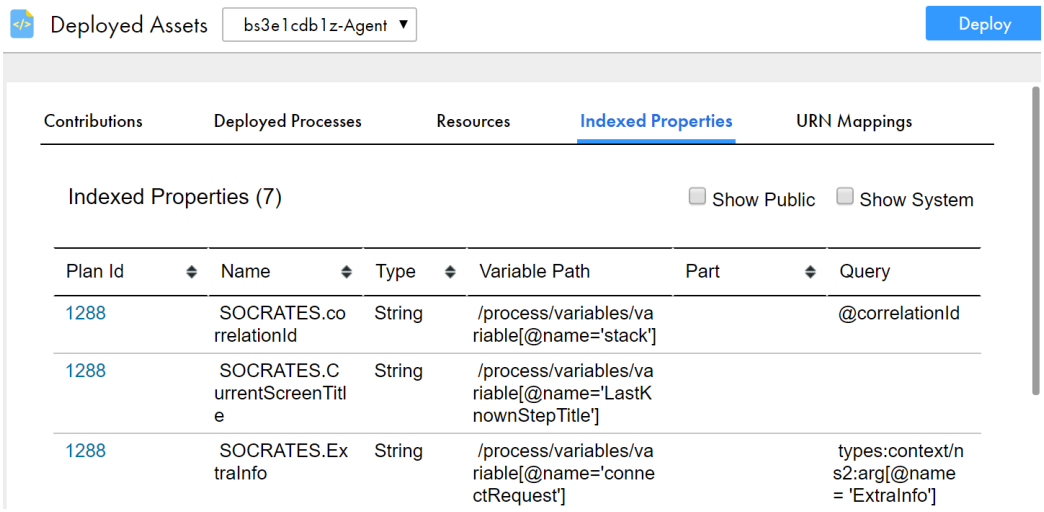
- Select a State—All, Online or Offline. For details, see *Contributions*.
- Select a resource Type from the list.
- Type in a Resource name—You can use wildcards, such as `tutor*.*`
- Type in a Target Namespace—You can use wildcards.
- Type in a Group name—If a group name was defined during deployment, you can search for it.

Select **Submit** to display the filtered list of resources.

Indexed Properties

An indexed property is a variable property that serves as a selection filter for active processes. It can also be used in defining events for business event processing. This property holds a piece of data, such as a customer Id, application date, or amount. Using an indexed property in an event service allows for fine-grained analysis of business events as they are monitored by the Event Manager. Using an indexed property in a selection query provides a fast way to filter processes based on important data items.

The following image shows a sample **Index Properties** tab .



For example, you can retrieve a list of faulting processes that share the same indexed property, suspend processes, fix bad data values, and continue process execution.

Indexed properties are defined in the process deployment descriptor file. Information in this table is as follows:

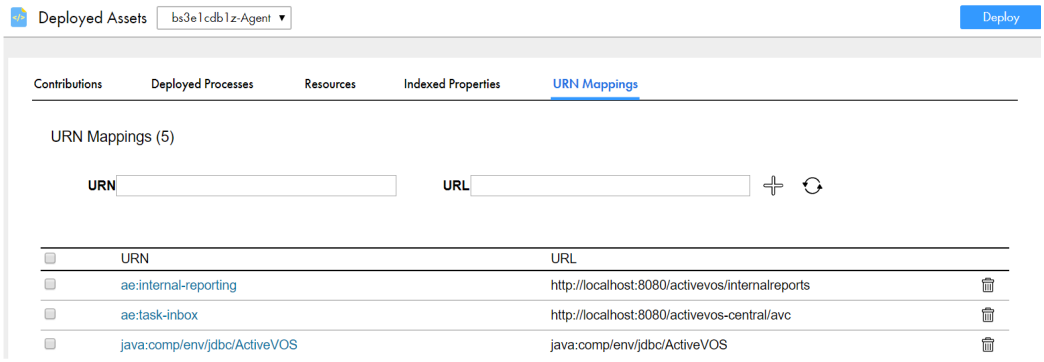
Item	Description
Plan ID	The deployed process associated with the indexed property; clicking on this link displays the "Deployed Process Version Detail Page" on page 89page.
Name	Indexed property name. This name appears in the Indexed Property list in the selection filters Expression Builder.
Type	Schema type, such as string or double
Variable Path	Process variable name and declaration location in the process
Part	Process variable part for message type variables
Query	Process variable part detail (optional)

URN Mappings

On the URN Mappings tab , there are system mappings and user-defined mappings.

You see four default mappings if you select a Secure Agent.

The following image shows a sample URN Mappings tab :



System Mappings

Process Server provides the following default URN Mappings for system services. You may need to manually add a mapping if it wasn't part of your initial configuration.

URN	URL
`ae:internal-reporting`	`http://localhost:8080/activevos/internalreports` Default address of the BIRT reporting engine for a deployed process containing a reporting service.
`ae:task-inbox`	`http://localhost:8080/activevos-central/avc` Default address of Process Central. Be sure to change the `host:port` to match your installed location, if needed.
j`java:comp/env/jdbc/ActiveVOS`	This value is created using your specified JNDI database name
`urn:aeHostEnvironmentRuntime`	`avHostEnvironmentRuntimeAccess`

User-Defined Mappings

You can assign a physical address to a universal resource name (URN). The URN is a logical address of a partner link, specified in a deployment resource.

URN mappings provide a flexible and dynamic way to define target endpoint references. Use URN mappings to specify the physical address of a partner link endpoint reference instead of using the address specified in a process deployment descriptor (`.pdd`) file or WSDL file. By mapping a URN to a URL, you do not have to rely on invoking a statically defined endpoint address. URN mappings give you flexibility, for example, to deploy the same BPR files for testing and production environments.

Also, if you specify a URL, you can replace the URL by mapping it to a different URL.

The following example illustrates one type of URN to URL mapping:

```
urn:localhost = http://localhost:8080/active-bpel/services/${urn.3}
```

This mapping might be used when a process is deployed with the following partner link address information:

```
<partnerLink name="assessor">
   <partnerRole endpointReference="static"
     invokeHandler="default:Address">
      <wsa:EndpointReference xmlns:assessor="http://
       tempuri.org/services/loanassessor">
      <wsa:Address>urn:localhost:AssessRisk</wsa:Address>
      <wsa:ServiceName PortName=
        "SOAPPort">assessor:LoanAssessor</wsa:ServiceName>
   </partnerRole>
</partnerLink>
```

The Process Server invocation framework resolves the URN as follows:

```
urn:localhost:AssessRisk = http://localhost:8080/active-bpel/services/AssessRisk
```

Here are some ways you can map URNs to URLs. Note that each segment of the URN is separated by a colon.

URN	URL
urnSegment1:urnSegment2	http://localhost:8080/active-bpel/services/MyService
http://ServerA:8080/active-bpel/services/MyService	http://ServerB:8081/active-bpel/services/MyService
urn:localhost:service	http://localhost:${AE-NODE1-PORT}/active-bpel/services/${urn.4}

The last example in the table above shows how you can use variable substitution in an URL.

The URL values can optionally contain variables. The variables can be environment variables accessible through `java.lang.System.getProperties()` or a segment from the URN itself. The Apache Ant style variable declaration of `${property}` is used to identify a property within the URL. Segments from the input URN value can be referenced by using a special property naming convention of `${urn.offset}` where offset is a one-based offset identifying the segment from the input URN value to use for substitution.

The URL in the third mapping in this table contains two variables. The `${AE-NODE1-PORT}` variable pulls the port number from an environment variable. This variable would need to be set as a -D parameter on the Java runtime environment (for example, `java -D AE-NODE1-PORT =8080 ...`) or populated externally to the Process Server.

The `${urn.4}` variable, also in this example, references the fourth segment from the input URN value. Notice that the URN contains only three segments. The URN in the `.pdd` file should contain at least one other segment. A sample URN might be:

`urn:localhost:service:StoreService`.

As the value of the fourth segment of this URN is `StoreService`, the resulting URL is:

```
http://localhost:8080/active-bpel/services/StoreService/
```

Managing URN Mappings

Operations you can perform are:

- Add a new URN, fill in the URN and URL fields within the Add new URN Mapping section and then press the **Add/Update Mapping** button.
- Edit an existing URN, click on its link in the top area of this tab . Process Server fills in the URN and URL fields within the Add new URN Mapping section. After making changes, press the **Add/Update Mapping** button.

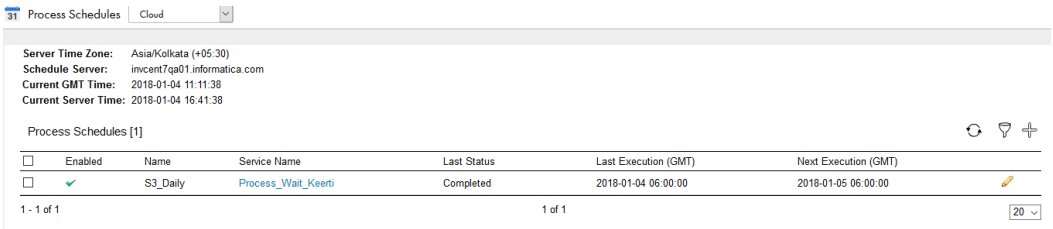
- To delete a mapping, select the check box next to the mapping and select **Delete**.

CHAPTER 4

Process Schedules

A BPEL process may be created to perform routine maintenance or another recurring task. If desired, you can deploy the process and then set a schedule to run it, such as weekly or on a specific day of the month. You can set a fine-grained schedule (intraday, seconds), if desired.

The following image shows the Process Schedules page:



The following table describes information you'll see on this page.

Field	Explanation
Enabled	A checkmark next to a schedule name indicates the process is enabled for execution
Name	Descriptive name of the schedule. For example, use the process name, with or without other descriptive details
Service Name	A service from the **Service Definitions** list. Select the Service Name to view the service definition. Hover help displays the process name associated with the service.
Last Status	Execution state of the active process, such as Completed or Running. Select the Status to view the Active Process Details page. Hover help displays the Process Id.
Last Execution	Date and time of the last process execution, if any
Next Execution	Date and time of the next scheduled execution, if the schedule is enabled

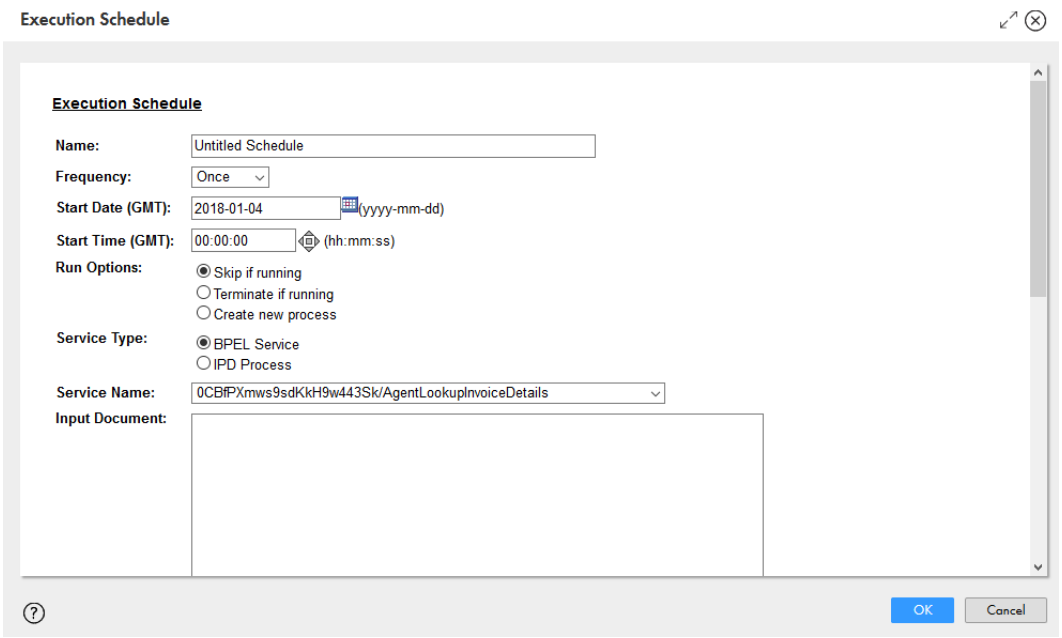
In a clustered environment, one of the nodes acts as a scheduling master at any time. Scheduled processes are elected to run only on the schedule master node. The schedule node is elected based on the server that starts up first. Only if there is a drop in the Secure Agent cluster communications do other nodes choose to get elected as the schedule master. Also Process Server does not maintain a history of the execution of scheduled processes.

To create a scheduled process, click the **Create Schedule** link at the top of the page. Process Server displays a dialog box that is described in *Creating a New Process Schedule*.

See also *Managing and Modifying Execution Schedules*.

Creating a New Process Schedule

To create a scheduled process, click **Add** at the top of the **Process Schedule** page. You see the following dialog box:



Execution Schedule	
Name	Provide a name to appear in the schedule list.
Frequency	- **Once**. Select this to execute on the start date and time specified. **Tip:** This option is good for running a test execution. You can then duplicate the details for a different execution schedule. - **Intraday**. Multiple execution times within a day. - **Daily**, **Weekly**, or **Monthly** allow for fine-grained control of the frequency schedule
Start Date on Server	For one-time (Frequency = Once) execution, type in a date, or select the Date Chooser to enter a date. In the Date Chooser, select Now to enter the current day.
Start Time on Server	The default is midnight (00:00:00). Type in a time of day, or select the Date Chooser to enter a time. In the Date Chooser, select Now to enter the current time.
Schedule Options	- For **Intraday**, select a period of hours, minutes or seconds. Optionally enable the start and end hours. - For **Daily**, select *Every Day* (default), *Every Weekday* (M-F), or *Every n days* - For **Weekly**, select one or more days of the week. For example, select Monday to execute every Monday. Select Monday and Thursday to execute every Monday and Thursday. Optionally, specify a weekly execution frequency of 1 to n, where n schedules every 2nd, 3rd, or nth week, such as 12th week. - For **Monthly**, select a specific day, or select the first, second, third, fourth, or last day of the month. For example, select Last Monday to execute on the last Monday of the month. **Tip:** To execute on the last day of each month, enter a Day of **31**. The correct last day of the month is adjusted. Select the months individually, if desired. By default, all months are selected.

Execution Schedule	
Run Options	Select from the following options: - **Skip if previous still running** (*default*). Does not schedule another process instance if the previous instance did not complete. - **Terminate previous if still running**. Terminates a previously scheduled execution prior to starting a new instance. - **Always create new process** . Schedules a new process execution regardless of the status of any prior executions.
Service Type	Choose whether the service is BPEL or IPD-based.
Service Name/ Select Service	If the service type is BPEL, type in a service name. Open the **Service Definitions** list to view names, then type in just the service name (without any extensions), such as `myScheduledService`. If the service type is IPD, select one of the services in the picklist.
Input Document/ POST Body	If the process is implemented using Process Developer and of BPEL type, follow either of the below steps: - If the process implements a SOAP interface, enter the SOAP input message data contained within the SOAP Body Element, that is, the XML input message used to invoke the process. The data must conform to the WSDL interface of the process. - If the process implements a REST interface, you first need to create a SOAP wrapper for the REST service. Then, follow the instructions above. If you use Process Designer to implement the process and the process is of the Informatica Process Definition (IPD) type, enter the JSON body of the POST message used to invoke the process.

After you enter this information, Process Server creates a new scheduled process and you see that it appears in the Scheduled Process page.

See also *Managing and Modifying Execution Schedules*.

Managing and Modifying Execution Schedules

Use the icon, check box, and drop-down list of the Process Schedules list to manage the list.

Field	Explanation
Pencil icon (Edit)	Select the pencil icon edit the schedule
Select/Select All	Select an individual execution schedule or select the Select All check box to run, duplicate, delete, enable, or disable the schedule
Run Now	Execute the selected processes
Duplicate	Duplicate selected schedules. Initially the new schedules are disabled.
Delete	Delete the selected execution schedules, including all details
Enable/Disable	Enable or disable the selected execution schedules. If you enable a schedule, the process executes at the next execution time. **Tip:** A disabled schedule may still be executed immediately (Run Now), duplicated, or deleted.

See also "Creating a New Process Schedule" on page 57.

Process scheduler REST APIs

Application Integration includes process scheduler REST APIs to manage scheduled processes.

You can use REST APIs to view, create, update, and delete process schedules. The APIs support basic authentication and session ID authorization.

Get Schedule API

You can use the Get Schedule API to request details about the process schedules in the organization.

If the REST API call is successful, the API returns the status code 200 in the response body. For more information about REST API responses, see *Data Integration REST API Reference*.

You can use the Get Schedule API to perform the following tasks:

APIs	Description
`GET <Informatica Cloud Application Integration URL>/active-bpel/restadmin/scheduler/schedules/?tenantContextId=<organization ID>`	View a list of process schedules that run on the Cloud Server within a specific organization.
`GET <Informatica Cloud Application Integration URL>/active-bpel/restadmin/scheduler/schedules/?tenantContextId=<organization ID>&name=<scheduleName>`	View a specific process schedule that runs on the Cloud Server within a specific organization.
`GET http://<agentName>:<httpPortNumber>/process-engine/restadmin/scheduler/schedules`	View a list of process schedules that run on a specific Secure Agent.
`GET http://<agentName>:<httpPortNumber>/process-engine/restadmin/scheduler/schedules?name=<scheduleName>`	View details of a specific process schedule that runs on a specific Secure Agent.

The following example shows a sample response when you view the process schedules:

```
{
"totalRowCount": 1,
"completeRowCount": true,
"schedulerInitialized": true,
"scheduledItems": {
"scheduledItem": [
{
"tenantContextId": "fDe0qZ1yc48bB2mRUmUYIN",
"name": "ProcessScheduleTest",
"ipdService": true,
"once": null,
"intraday": {
"executeEvery": {
"value": 30,
"interval": "SECONDS"
},
"onlyBetween": {
"betweenStart": "2021-09-17T06:38:25.000+00:00",
```

```
"betweenEnd": "2021-09-17T06:39:35.000+00:00"
}
},
"daily": null,
"weekly": null,
"monthly": null,
"runOptions": "SKIP",
"serviceName": "fDe0qZ1yc48bB2mRUmUYIN/ProcessSchedule",
"inputDocument": null,
"scheduleId": "623407279198179328",
"enabled": true,
"lastProcessId": "0",
"lastExecution": null,
"nextExecution": "2021-09-17T06:38:27.706+00:00",
"planId": "623407273670086656"
}
]
}
}
```


Create Schedule API

You can use the Create Schedule API to create a process schedule that runs on the Cloud Server or on Secure Agents.

If the REST API call is successful, the API returns the status code 204 in the response body. For more information about REST API responses, see *Data Integration REST API Reference*.

You can use the Create Schedule API to perform the following tasks:

APIs	Description
`POST <Informatica Cloud Application Integration URL>/active-bpel/restadmin/scheduler/schedules`	Create a process schedule based on an Informatica Process Designer (IPD) service. Set `ipdService` to `true` to create a process schedule based on a BPEL service. The following snippet shows a sample payload when you make an API request: `{ "tenantContextId": "9kCHfynWgUBfPh8D1FcYyL", "name": "APICreatedSchedule5", "ipdService": true, "once": null, "intraday": { "executeEvery": { "value": 30, "interval": "SECONDS" }, "onlyBetween": null }, "daily": null, "weekly": null, "monthly": null, "runOptions": "SKIP", "serviceName": "Process6-16", "inputDocument": null }` **Note:** The serviceName is the API name of the process.
`POST <Informatica Cloud Application Integration URL>/active-bpel/restadmin/scheduler/schedules`	Create a process schedule based on a Business Process Execution Language (BPEL) service. Set `ipdService` to `false` to create a process schedule based on a BPEL service. The following snippet shows a sample payload when you make an API request: `{ "tenantContextId": "9kCHfynWgUBfPh8D1FcYyL", "name": "APIBpelSchedule2", "ipdService": false, "once": null, "intraday": { "executeEvery": { "value": 30, "interval": "SECONDS" }, "onlyBetween": null }, "daily": null, "weekly": null, "monthly": null, "runOptions": "SKIP", "serviceName": "9kCHfynWgUBfPh8D1FcYyL/Process6-16", "inputDocument": { "any": "<spi:initiate xmlns:spi=\"http://schemas.informatica.com/spi/types/2013/12/spi-interface/\" xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:avos=\"http://`

APIs	Description
	schemas.active-endpoints.com/appmodules/screenflow/2010/10/avosScreenflow.xsd\">\n\t\t\t</spi:initiate>", "otherAttributes": {} } } **Note:** The serviceName is the API name of the process.

POST http://<agentName>:<httpPortNumber>/process-engine/restadmin/scheduler/schedules	Create a process schedule that runs on a specific Secure Agent. You must include the agent name in the request.
	The following snippet shows a sample payload when you make an API request:
	{
	"tenantContextId": "$public",
	"name": "agentSch1",
	"ipdService": true,
	"once": null,
	"intraday": {
	"executeEvery": {
	"value": 30,
	"interval": "SECONDS"
	},
	"onlyBetween": null
	},
	"daily": null,
	"weekly": null,
	"monthly": null,
	"runOptions": "SKIP",
	"serviceName": "ProcessScheduledAgent",
	"inputDocument": null
	}
	Note: The serviceName is the API name of the process.

Update Schedule API

You can use the Update Schedule API to update a process schedule that runs on the Cloud Server or on Secure Agents.

If the REST API call is successful, the API returns the status code 204 in the response body. For more information about REST API responses, see *Data Integration REST API Reference*.

You can use the Update Schedule API to perform the following tasks:

APIs	Description

`PUT <Informatica Cloud Application Integration URL>/active-bpel/restadmin/scheduler/schedules`	Update a process schedule that runs on the Cloud Server.
	The following snippet shows a sample payload when you make an API request:
	`https://<Cloud Application Integration POD URL>/active-bpel/restadmin/scheduler/schedules`
	`{ "scheduleId": "<scheduleId>", "name": "<scheduleName>", "runOptions": "SKIP", "serviceName": "<process API Name>", "ipdService": "true", "tenantContextId": "<orgId>", "enabled": true, "once": null, "intraday": { "executeEvery": { "value": 1, "interval": "MINUTES" }, "onlyBetween": null }, "inputDocument": null }`

`PUT http://<agentName>:<httpPortNumber>/process-engine/restadmin/scheduler/schedules`	Update a process schedule that runs on a specific Secure Agent.
	`https://<Cloud Application Integration POD URL>/active-bpel/restadmin/scheduler/schedules`
	`{ "scheduleId": "<scheduleId>", "name": "<scheduleName>", "runOptions": "SKIP", "serviceName": "<process API Name>", "ipdService": "true", "tenantContextId": "$public", "enabled": true, "once": null, "intraday": { "executeEvery": { "value": 1, "interval": "MINUTES" }, "onlyBetween": null }, "inputDocument": null }`

Delete Schedule API

You can use the Delete Schedule API to delete all process schedules on the Cloud Server or on Secure Agents.

If the REST API call is successful, the API returns the status code 204 in the response body. For more information about REST API responses, see *Data Integration REST API Reference*.

You can use the Delete Schedule API to perform the following tasks:

APIs	Description

`DELETE <Informatica Cloud Application Integration URL>/active-bpel/restadmin/scheduler/schedules`	Delete a process schedule that runs on the Cloud Server.
	The following snippet shows a sample payload when you make an API request:
	`{`
	`"scheduleId": [926347078765268992]`
	`}`

`DELETE http://<agentName>:<httpPortNumber>/process-engine/restadmin/scheduler/schedules`	Delete a process schedule that runs on a specific Secure Agent.
	The following snippet shows a sample payload when you make an API request:
	`{`
	`"scheduleId": [104]`
	`}`

You can pass more than one schedule ID that are separated by commas. If even one schedule ID is invalid, none of the process schedules will be deleted.

CHAPTER 5

Processes

The **Processes** page shows a list of process instances that are running or have been run in the organization, along with the process version. States can be Running, Suspended, Completed, Compensatable (for a subprocess), or Faulted.

Use the Search feature to quickly find an active process instance. If you enter a Process ID, you see the process instance.

The following table shows the different process states:

Process State	Meaning
Compensatable	A sub-process is complete and eligible for compensation.
Completed	Normal completions.
Faulted	Processes completed with a fault.
Running	Normally running processes.
Suspended	A process suspended for any reason.

You can select a group of running, suspended, faulted, or completed processes and take some action on them.

On the **Processes** page, you can perform the following tasks:

1. View a list of processes that you want to work with. You can use selection filters to create a customized list.
2. Select some or all processes displayed on the current page.
   You can select up to the maximum records displayed for each page. You cannot select processes on subsequent pages. You can set the number of records that you want to view for each page.
3. From the **Action** list, perform one of the following actions:
   - Terminate: Terminates running processes.
   - Suspend: Suspends running processes.
   - Resume: Resumes suspended processes.

     Note: When you resume a process, Application Integration takes a few seconds to resume the process and display the updated process status. If you check the process status while the process is being resumed, the process status is displayed as **Suspended**. You can ignore this status. After the process is resumed and starts running, the process status is refreshed.

- Restart: Restarts completed, faulted, or suspended processes. For more information, see "Process Restart" on page 95.
- Delete: Deletes faulted or completed processes.
- Retry: Retries a suspended faulting process. When you retry a process, the process resumes and retries the faulting activity.

  Note: If you retry a process and then immediately restart it, the following error occurs:

  ```
  The required logs are not available for the process instance <Instance ID> for the Tenant Context <Org ID>
  ```

  Wait until the retry action completes before restarting the process.

You can access the following pages from the **Process** page:

Process View Detail

Click a process ID to see the **Process View Detail** page. You can see a chronological list of process activities, the process as it appears in the designer, and detailed step information. You can also access the **Active Process Detail** page.

Process Version Listing

Click a process name to see the **Process Version Listing** page. See the target name space, the version name, version number, plan ID, online date, offline date, and deployed date.

Deployed Process Version Details

Click a process version number to see the **Deployed Process Version Details** page. You can see the process version life cycle, update a process version, see My Role and Partner Role details and many more on ths page.

Process filters

The **Processes** page displays a list of process instances that are running or have been run in the organization. You can apply filters to view the processes that you would like to monitor and analyze.

You can create the following types of process filters:

Simple process filter

You can create a simple process filter to filter processes based on a field, operator, and value.

Advanced process filter

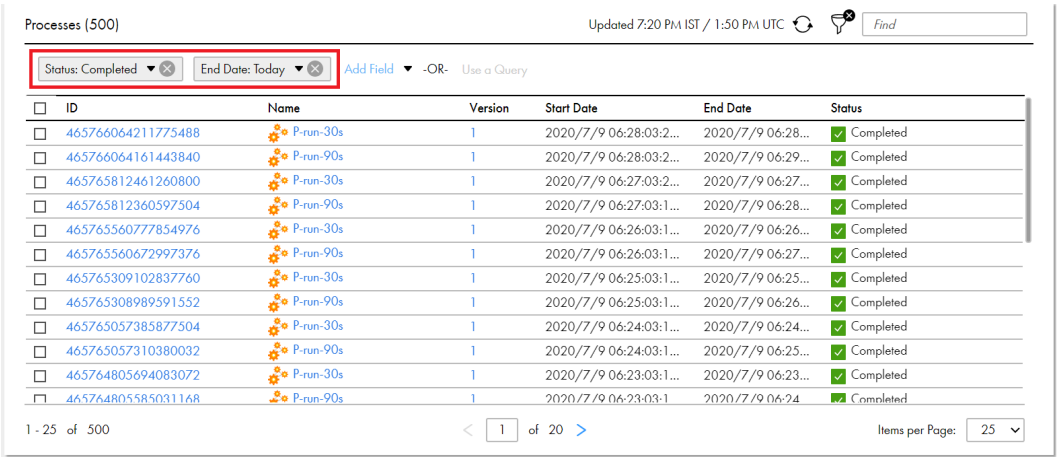
You can use a query to define an advanced process filter condition. Construct the query by using an extensive set of properties from pre-defined lists and by grouping expressions with logical operators.

Simple process filters

You can create a simple process filter to filter processes based on a field, operator, and value.

You can create one or more simple process filters. When you create multiple simple process filters, the Process Server joins all the filters with an AND operator and displays the processes that meet all the filter conditions.

For example, to view the processes that were completed today, you can create two simple process filter conditions with the **Status** field set to **Completed** and the **End Date** field set to **Today** as shown in the following image:

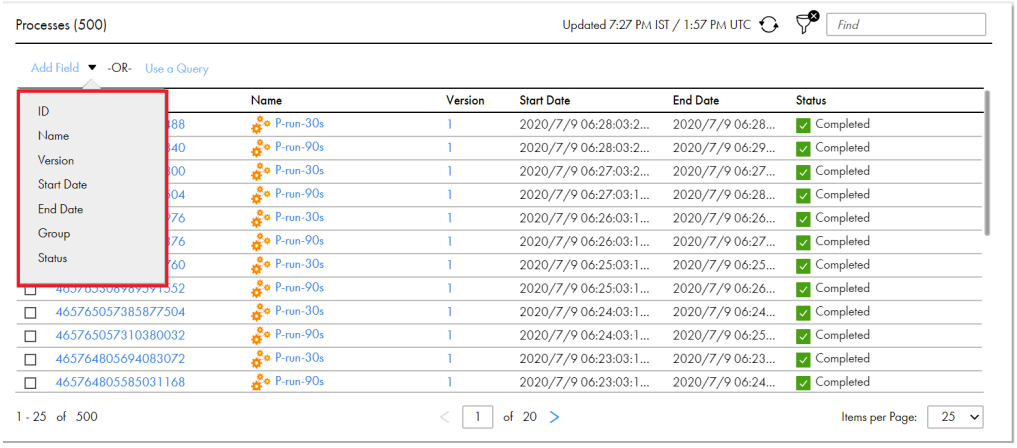


Configuring a simple process filter

1. In Application Integration Console, click **Processes** on the left navigation bar.
2. On the right pane, click the **Filter** icon.

   The **Add Field** and **Use a Query** options appear.
3. Click the down arrow adjacent to **Add Field** and select one of the following fields based on which you want to filter the processes:
   - ID
   - Name
   - Version
   - Start Date
   - End Date
   - Group
   - Status

   The following image shows the fields based on which you can filter processes:



4. Configure the filter condition based on the field that you selected.

The following table describes the steps that you must complete based on the field that you selected:

Field	Description
ID	The instance ID of the process. Every process run has a unique instance ID. If you selected **ID**, complete the following steps: 1. Select one of the following operators: • Equals • Does not equal • Less than • Less than or equal to • Greater than • Greater than or equal to • Between 2. Enter a value for the filter condition. If you selected the **Between** operator, enter two values to define a range based on which you want to filter the processes.
Version	The version number of the process. If you selected **Version**, complete the following steps: 1. Select one of the following operators: • Equals • Does not equal • Less than • Less than or equal to • Greater than • Greater than or equal to • Between 2. Enter a value for the filter condition. If you selected the **Between** operator, enter two values to define a range based on which you want to filter the processes.
Name	The name of the process. If you selected **Name**, enter a value for the filter condition.
Group	The group that a process belongs to. The group name is specified in the Process Deployment Descriptor (PDD) and displayed on the **Deployed Process Version Detail** page. If you selected **Group**, enter a value for the filter condition.
Start Date	The date when the process execution started. If you selected **Start Date**, select one of the following values based on which you want to filter the processes: • Any time • Today • Last Week • Last Month • Last Year • Custom Range If you selected **Custom Range**, click the calendar to specify a date and time range for the filter condition.

Field	Description
End Date	The date when the process completed, faulted, or was stopped or suspended. If you selected **End Date**, select one of the following values based on which you want to filter the processes: • Any time • Today • Last Week • Last Month • Last Year • Custom Range If you selected **Custom Range**, click the calendar to specify a date and time range for the filter condition.
Status	The state of the process execution. If you selected **Status**, select one of the following values based on which you want to filter the processes: • Completed • Faulted • Running • Suspended

5. Create additional filters if needed.

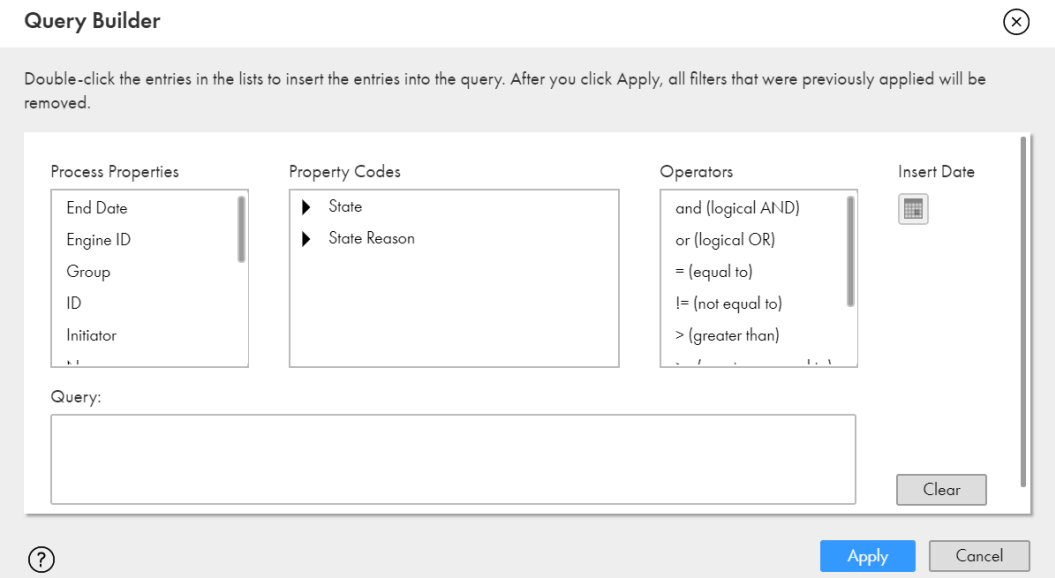
   The Process Server joins all the filters with an AND operator and displays the processes that meet all the filter conditions.

To remove a filter, click **Close** next to the filter condition. To remove all the filters and view all the processes, click **Close** next to the **Filter** icon.

Advanced process filters

You can construct a query to define an advanced process filter condition. You can construct the query by using an extensive set of properties from pre-defined lists and by grouping expressions with logical operators.

The following image shows the **Query Builder** page:



You can use the **Process Properties** and **Property Codes** lists to select the properties based on which you want to filter processes. Use the **Operators** list to define the operators that you want to include in the query.

The following table describes the properties that are available in the **Process Properties** list:

Process Property	Description	Sample Query
End Date	The date when the process completed, faulted, or was stopped or suspended. The **End Date** field uses Coordinated Universal Time (UTC). Double-click to add the **getProcessProperty("EndDate")** property to the query. Select an operator and click the calendar below the **Insert Date** field to select an end date. To use the **=** or **!=** operator with an end date in the query, you must specify the end date in milliseconds. Otherwise, the query returns incorrect results. You can also use a range of dates in the query to filter processes based on the end date.	Use one of the following queries to filter the processes whose end date is August 21, 2020: - `getProcessProperty("EndDate") = "1597994701818"` - `getProcessProperty("EndDate") > "2020/08/20" and getProcessProperty("EndDate") < "2020/08/22"`
Engine ID	The ID of the engine that runs the process. Double-click to add the **getProcessProperty("EngineId")** property to the query. Select an operator and type a value for the engine ID.	Use the following query to filter the processes whose engine ID is 1: `getProcessProperty("EngineId") = 1`

Process Property	Description	Sample Query
Group	The group that a process belongs to. The group name is specified in the Process Deployment Descriptor (PDD) and displayed on the **Deployed Process Version Detail** page. Double-click to add the **getProcessProperty("Group")** property to the query. Select an operator and type a value for the group name. Enclose the group name within single quotation marks (') or double quotation marks (").	Use the following query to filter the processes that belong to the North West Region group: `getProcessProperty("Group") = "North West Region"`
ID	The instance ID of the process. Every process run has a unique instance ID. Double-click to add the **getProcessProperty("Id")** property to the query. Select an operator and type a value for the ID.	Use the following query to filter the processes whose instance ID is greater than 434251505437396992: `getProcessProperty("Id") > 434251505437396992`
Initiator	The name of the user who initiated the process. Double-click to add the **getProcessProperty("Initiator")** property to the query. Select an operator and type a name for the initiator. Enclose the initiator name within single quotation marks (') or double quotation marks (").	Use the following query to filter the processes that were initiated by John: `getProcessProperty("Initiator") = 'John'`
Name	The name of the process. Double-click to add the **getProcessProperty("Name")** property to the query. Select an operator and type a process name. Enclose the process name within single quotation marks (') or double quotation marks (").	Use the following query to filter the processes whose name is LoanApproval: `getProcessProperty("Name") = "LoanApproval"`
Namespace	The target namespace of the process. The target namespace uses the following syntax: `urn:screenflow:process:<process_name>` The target namespace value is unique for each process. Double-click to add the **getProcessProperty("Namespace")** property to the query. Select an operator and type a target namespace. Enclose the target namespace within single quotation marks (') or double quotation marks (").	Use the following query to filter the process whose target namespace is urn:screenflow:process:LoanApproval: `getProcessProperty("Namespace") = 'urn:screenflow:process:LoanApproval'`

Process Property	Description	Sample Query
Start Date	The date when the process execution started. The **Start Date** field uses Coordinated Universal Time (UTC). Double-click to add the **getProcessProperty("StartDate")** property to the query. Select an operator and click the calendar below the **Insert Date** field to select a start date. To use the **=** or **!=** operator with a start date in the query, you must specify the start date in milliseconds. Otherwise, the query returns incorrect results. You can also use a range of dates in the query to filter processes based on the start date.	Use one of the following queries to filter the processes whose start date is August 21, 2020: - `getProcessProperty("StartDate") = "1597994701818"` - `getProcessProperty("StartDate") > "2020/08/20" and getProcessProperty("StartDate") < "2020/08/22"`
State	The state of the process execution. Double-click to add the **getProcessProperty("State")** property to the query. Select an operator. In the **Property Codes** list, expand the **State** list and double-click a state value to insert the associated state code into the query. The following list describes the state values and the associated state codes: - Running. 1. The process is running. - Suspended. 2. The process is suspended. - Completed. 3. The process has completed successfully. - Faulted. 4. The process completed with a fault. - Compensatable. 5. A sub-process is complete and is eligible for compensation.	Use the following query to filter the processes that are in the faulted state: `getProcessProperty("State") = "4"`

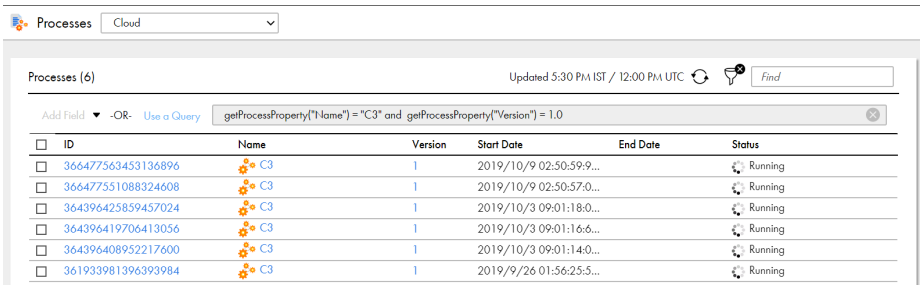
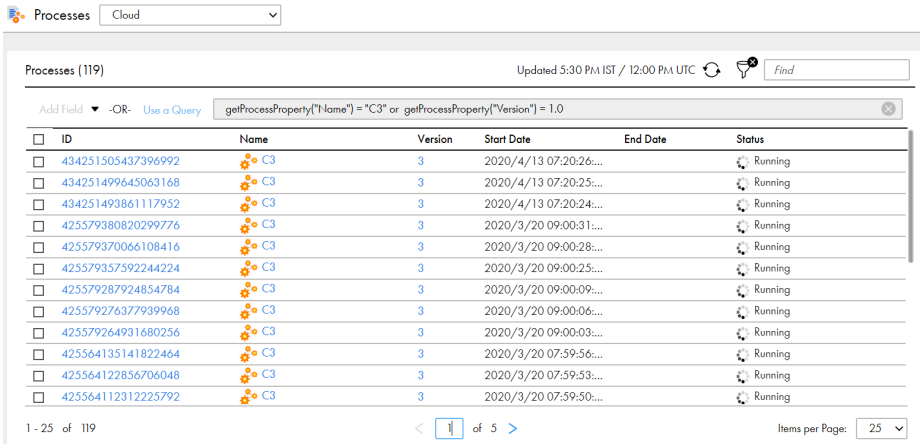
Process Property	Description	Sample Query
State Reason	The reason why a process is in the suspended state. Double-click to add the **getProcessProperty("StateReason")** property to the query. Select an operator. In the **Property Codes** list, expand the **State Reason** list and double-click a state reason value to insert the associated state reason code into the query. The following list describes the state reason values and the associated state reason codes: - Suspended (Manual). 0. Applies to processes that were suspended manually. - Suspended (Faulting). 1. Applies to processes that were suspended due to a fault that was not caught. - Suspended (Activity). 2. Applies to processes that were suspended at a BPEL suspend activity. - Suspended (Migration Error). 7. Applies to processes that were suspended due to a migration error for a process that has been migrated to a new process version. - Suspended (Migration Warning). 8. Applies to processes that were suspended due to a migration warning for a process that has been migrated to a new process version. Warnings indicate differences found in the new version that might need review and updates before you resume the process. - Pre Migration Copy. 20. Applies to an old process instance that has been saved temporarily until the new process instance has been resumed or terminated.	Use the following query to filter the processes that are in the suspended state because they faulted: `getProcessProperty("StateReason") = 1`
Tenant ID	The tenant identification number for the process. The tenant identification number is unique for each organization. Double-click to add the **getProcessProperty("TenantId")** property to the query. Select an operator and type a value for the tenant ID. Enclose the tenant ID within single quotation marks (') or double quotation marks (").	Use the following query to filter the processes whose tenant ID is 3i5mW6m6816cHjNTPH8LwE: `getProcessProperty("TenantId") = "3i5mW6m6816cHjNTPH8LwE"`

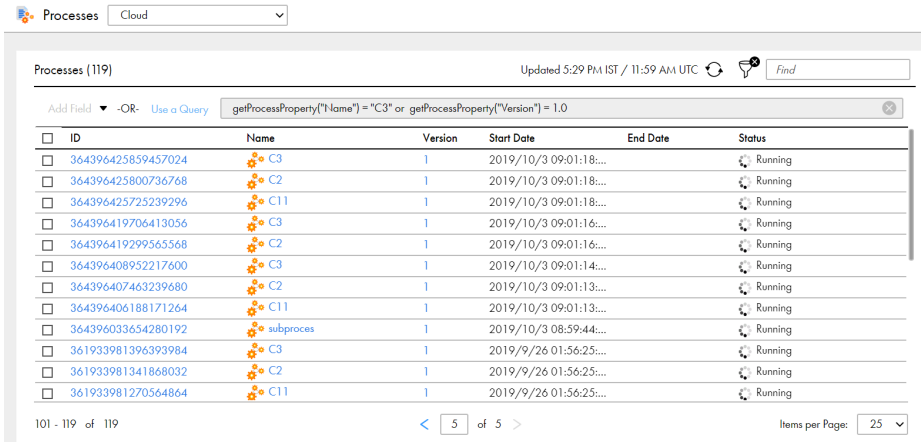
Process Property	Description	Sample Query
Title	The name of the process. Double-click to add the **getProcessProperty("Title")** property to the query. Select an operator and type a process name. Enclose the process name within single quotation marks (') or double quotation marks (").	Use the following query to filter the processes whose name is LoanApproval: `getProcessProperty("Name") = "LoanApproval"`
Version	The version number of the process. Double-click to add the **getProcessProperty("Version")** property to the query. Select an operator and type a value for the version number.	Use the following query to filter the processes whose version number is equal to 1: `getProcessProperty("Version") = 1`

The following table describes the properties that are available in the **Property Codes** list:

Property Code	Description
State	The state of the process execution. Expand the **State** list and double-click a state value to insert the associated state code into the query. The following list describes the state values and the associated state codes: - Running. 1. The process is running. - Suspended. 2. The process is suspended. - Completed. 3. The process has completed successfully. - Faulted. 4. The process completed with a fault. - Compensatable. 5. A sub-process is complete and is eligible for compensation.
State Reason	The reason why a process is in the suspended state. Expand the **State Reason** list and double-click a state reason value to insert the associated state reason code into the query. The following list describes the state reason values and the associated state reason codes: - Suspended (Manual). 0. Applies to processes that were suspended manually. - Suspended (Faulting). 1. Applies to processes that were suspended due to a fault that was not caught. - Suspended (Activity). 2. Applies to processes that were suspended at a BPEL suspend activity. - Suspended (Migration Error). 7. Applies to processes that were suspended due to a migration error for a process that has been migrated to a new process version. - Suspended (Migration Warning). 8. Applies to processes that were suspended due to a migration warning for a process that has been migrated to a new process version. Warnings indicate differences found in the new version that might need review and updates before you resume the process. - Pre Migration Copy. 20. Applies to an old process instance that has been saved temporarily until the new process instance has been resumed or terminated.

The following table describes the operators that you can use in the query:

Operator	Description
and	logical AND When you join two conditions with an `and` operator, the Process Server displays processes that meet both the conditions. For example, consider the following query: `getProcessProperty("Name") = "C3" and getProcessProperty("Version") = 1` This query returns processes whose name is C3 and whose version number is 1 as shown in the following image:  **Note:** The **and** operator is case sensitive.
or	logical OR When you join two conditions with an `or` operator, the Process Server displays processes that meet either of the two conditions. For example, consider the following query: `getProcessProperty("Name") = "C3" or getProcessProperty("Version") = 1` This query returns processes whose name is C3 as shown in the following image:  The query also returns processes whose version number is 1 as shown in the following image:

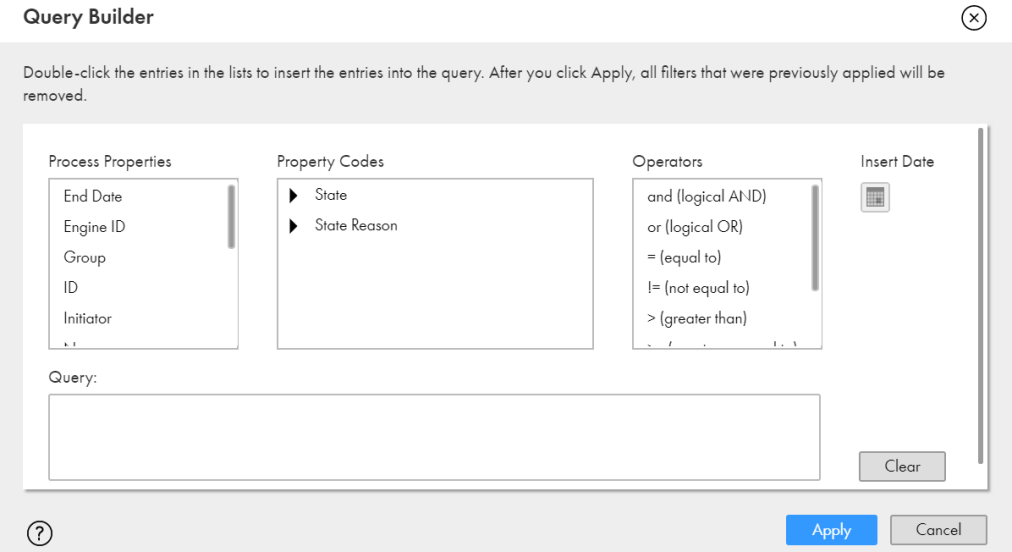
Operator	Description
	 **Note:** The **or** operator is case sensitive.
=	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Lesser than or equal to

Configuring an advanced process filter

1. In Application Integration Console, click **Processes** on the left navigation bar.
2. On the right pane, click the **Filter** icon.

   The **Add Field** and **Use a Query** options appear.
3. Click **Use a Query**.

   The **Query Builder** page appears.
   The following image shows the **Query Builder** page:



4. Double-click the entries in the **Process Properties**, **Property Codes**, and **Operators** lists to insert the entries into the query. Optionally, edit the query or click **Clear** to clear the query.

   The query that you configure appears in the **Query** box.

5. Click **Apply**.

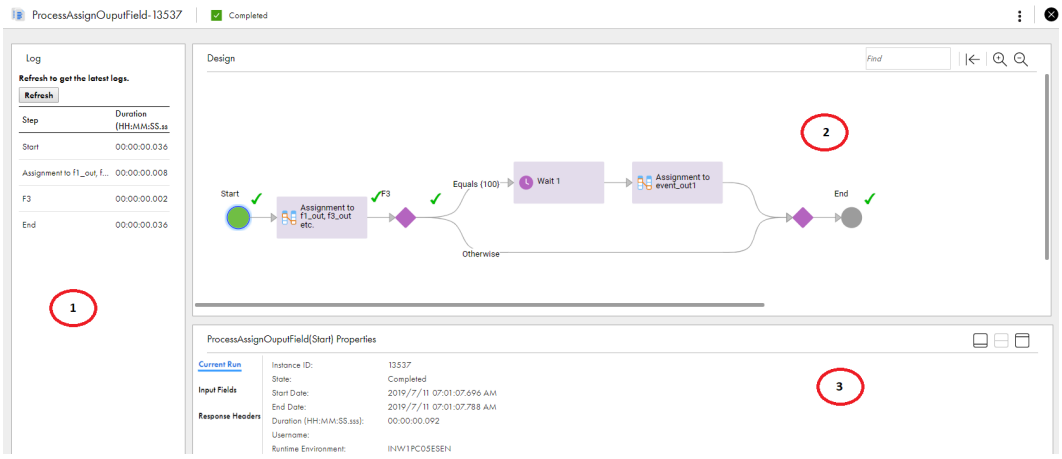
   The Process Server removes all filters that were previously applied and displays the processes that meet the configured query.

To remove the filters and view all the processes, click **Close** next to the filter condition or click **Close** next to the **Filter** icon.

Process View Detail Page

Click a process ID on the Process page to see the **Process View Detail** page.

The following image shows a sample **Process View Detail** page for a process created using Application Integration:



The **Process View Detail** page contains the following sections:

- Section **1** shows the **Log** area. You see a chronological list of process activities. As you navigate through the Log list and click a step, Application Integration Console highlights the corresponding step on the right.
- Section **2** shows the process as you see it in Process Designer.
- Section **3** displays information about the highlighted step. Depending on the step selected, this section might show fault information, other information, or is empty.
  Click **Advanced View** in the top right corner to see the **Active Process Detail** page of this process.

Note: To view runtime information for an Assignment step of a process, use one of the following options:

- Set `X-Debug=true` as the HTTP header in the request.
- Use the **Run Using** option to create process inputs and run a process with one or more process inputs.

Start Step Properties

The Start step properties display information about the current run of a process.

When you click the Start step, the properties panel displays the following information:

Property	Description
Current Run	You see the following properties: - Instance ID. A unique number that identifies the current process instance. - Status. The state that the process instance is in. Processes can be in the Running, Completed, Faulted, or Suspended states. - Start Date. The date and time at which the process instance was started. - End Date. The date and time at which the process instance ended. You see an End Date value if a process is in the Completed or Faulted states. - Duration. The amount of time the process instance ran before it completed. You see a Duration value if a process is in the Completed or Faulted states. - Username. The name of the user who ran this instance of the process. - Runtime Environment. The runtime environment in which the process ran.
Input Fields	Displays the names and values of the input fields that you configured in the process. For Kafka connections, Kafka consumers can consume headers sent with Kafka messages. The message headers are displayed in the **event** input field and can be used in a process.
Response Headers	Displays the header names and response values for the response headers that you configured in the process.

Fault Details

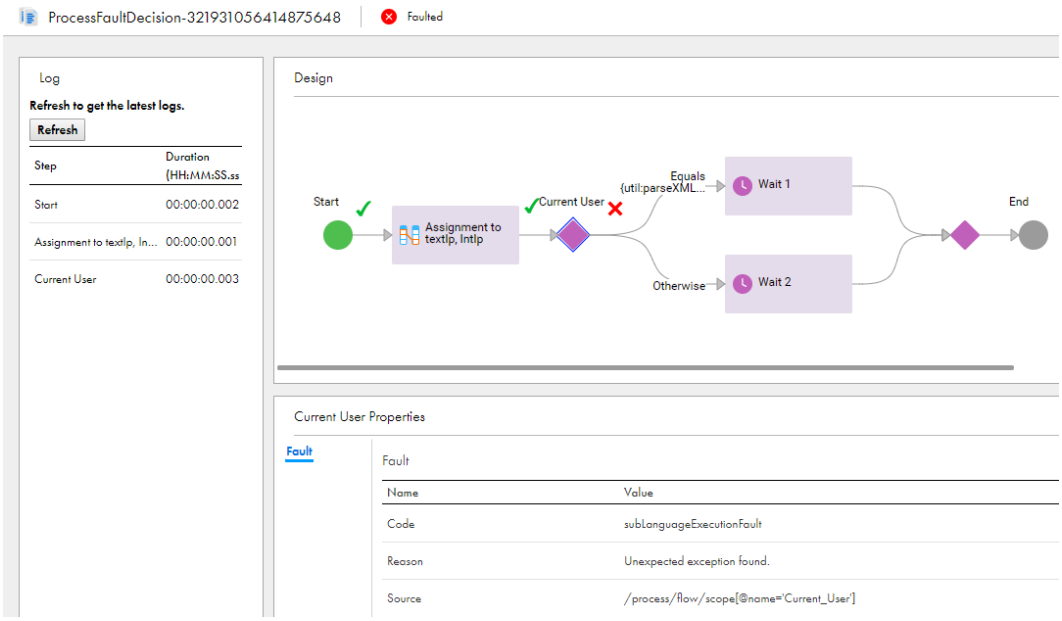
When a process faults, the Start step displays the overall fault information for the overall process. The **Process View Detail** page displays fault information for the steps that faulted.

Click a faulted step and click the **Fault** tab to view the fault information. The **Fault** tab displays a code, reason, source, and details for the fault that help you resolve the faults.

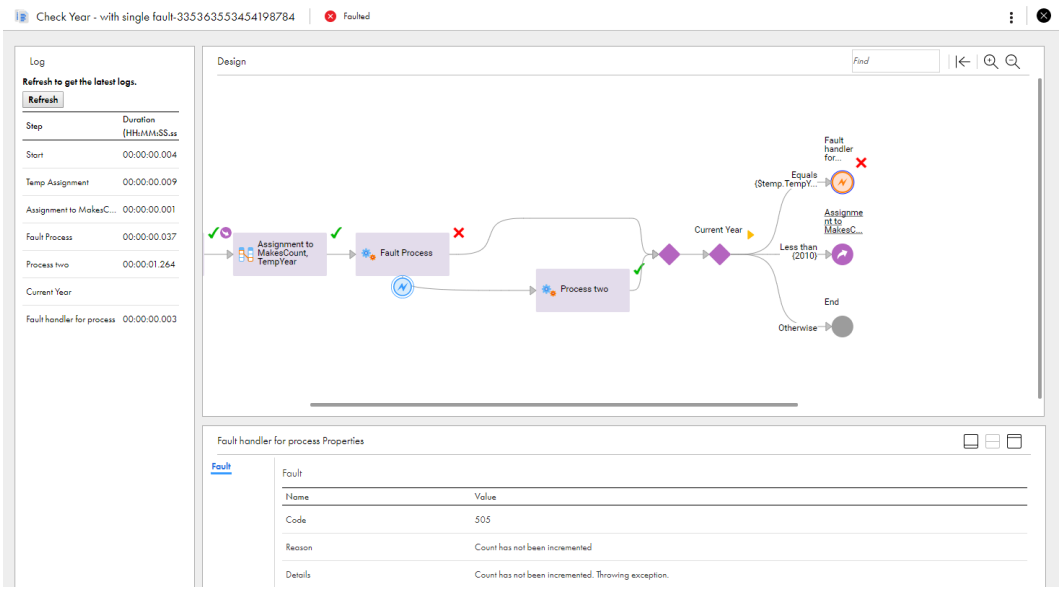
The following table describes the fault information that is displayed in the **Fault** tab:

Detail	Description
Code	The code that categorizes the fault. Here are some sample codes: - AeAgentTunnelHttpException - AeInvokePrepareException - CreateDataException - DataProviderException - faultResponse - HTTP_500 - Invalid Data - runtimeError - subLanguageExecutionFault
Reason	The reason that the process faulted at a step.
Source	The BPEL step at which the error originated. **Note:** If you had configured fault handling for a step, the **Source** field is unavailable. Instead, the **Details** field displays the fault handling details that you configured for the step.
Details	Fault handling details that you configured for the step.

The following image shows the **Faults** tab for a process that faulted at a Decision step:



The following image shows the **Faults** tab for a Throw step that uses fault handling:



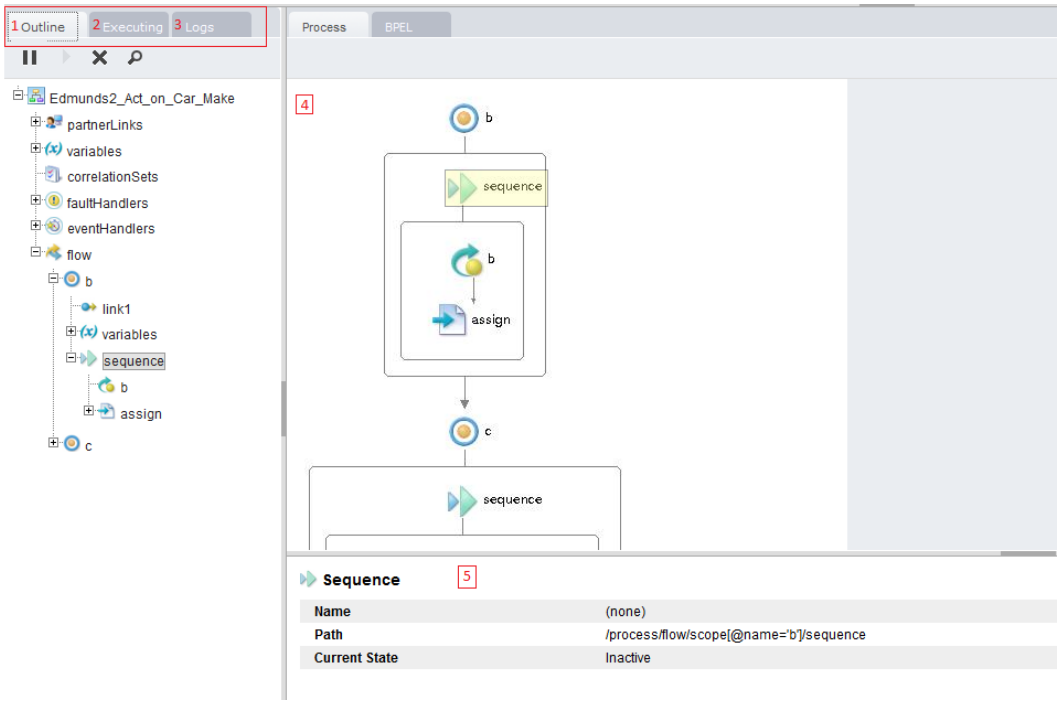
Notice that instead of the **Source** field, the **Details** field appears displaying the fault handling details that you configured for the Throw step.

Active Process Detail Page

When you select a process instance from the Processes page, the Active Process Detail page opens. You see many details about a process instance and you can suspend the process and perform process exception management.

Click **Advanced View** in the top right corner of the **Processes** to see the Active Process Detail page of a process.

The following image shows the page and indicates areas on it:



The following components are available:

	Area	What It Has
1	Outline view	Shows the structural elements of a BPEL process and the current process execution state of each activity. You can select an element to view its properties and values.
2	Executing view	Shows the current process execution state for a running process, stopping on the currently executing activity. If the process is completed, the execution view is blank.
3	Logs view	Shows execution details that you can use to analyze data changes, faults, and other details. You can also suspend a process and perform a process rewind of activities.
4	Process view	Shows the main process flow on a canvas. If the process has event, fault, and compensation handlers, you can view them by selecting the corresponding activity from the Outline. For example, select a Catch handler to view all fault handlers. You can also select an activity to view its properties. The BPEL tab shows the source XML code for the process.
5	Properties view	Appears for a selected element selected from the canvas any other component. For example, when the Process element is selected from the Outline view, you can see process properties and their current values, such as the current state and start date.

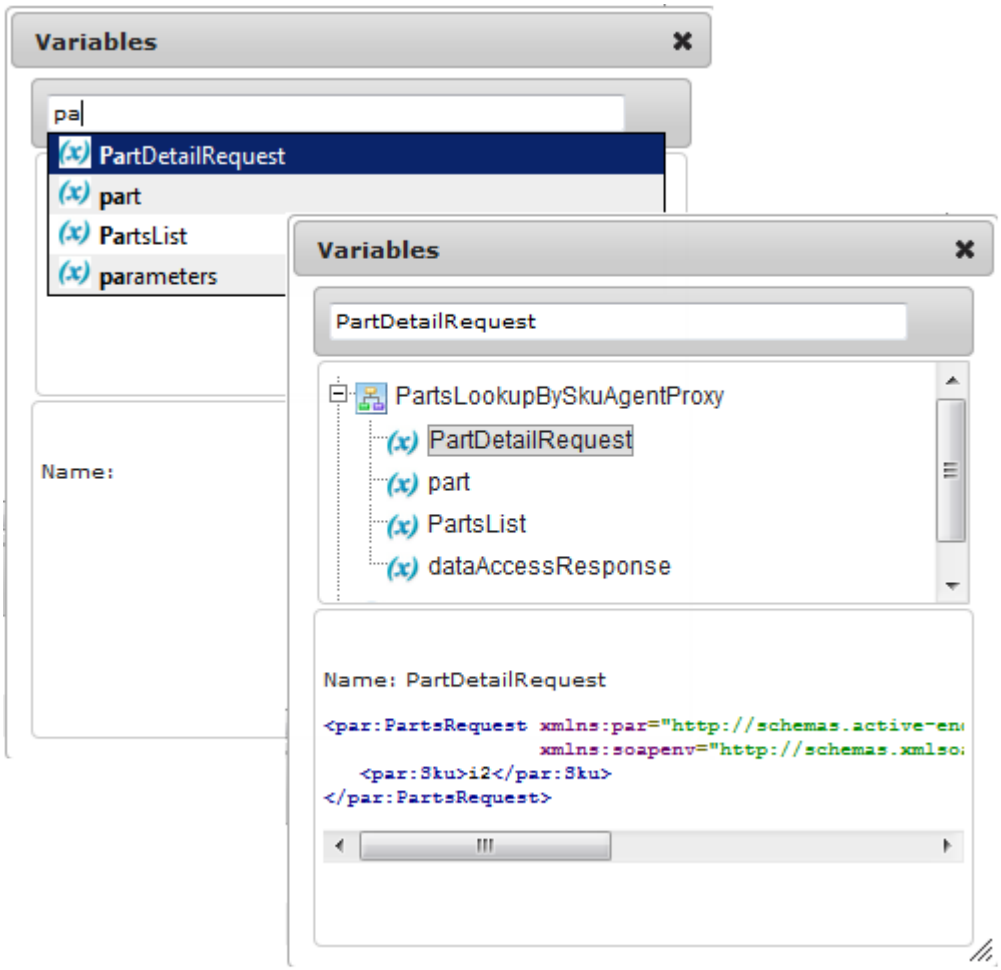
Additional windows open from the toolbar for Variables, Correlation Sets, Partner Links, and Debug threads for a convenient way to edit data and step/retry activities.

Toolbar

A toolbar allows you to select different views of this page and contains features for performing process exception management.

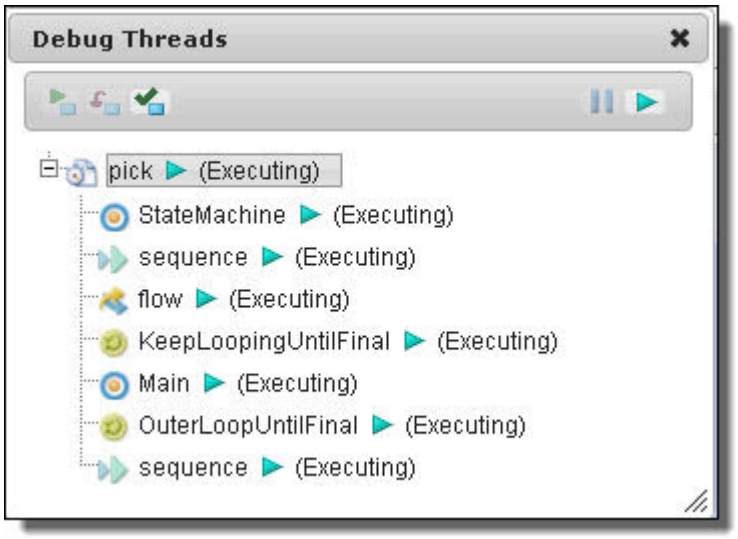
Top Row	The items on the top row let you select the default (all components), outline (outline + canvas), properties (properties + canvas), or canvas-only view. You can also display different views with URL view parameter with the following arguments: 1: Default, show all components (Outline, Canvas, Properties) 2: Canvas only 3: Outline and Canvas 4: Canvas and Properties
Bottom Row	Pressing one of these buttons brings up on of the following dialogs: 1. Variables 2. Partner Links 3. Correlation Sets 4. Debug Threads

The Variables, Partner Links, and Correlation Sets dialogs work similarly. Here, for example, is a Variables dialog:



As you type in a variables name, a picklist appears showing variables that match what you are typing (left dialog). After you choose one, the dialog displays information about the variable.

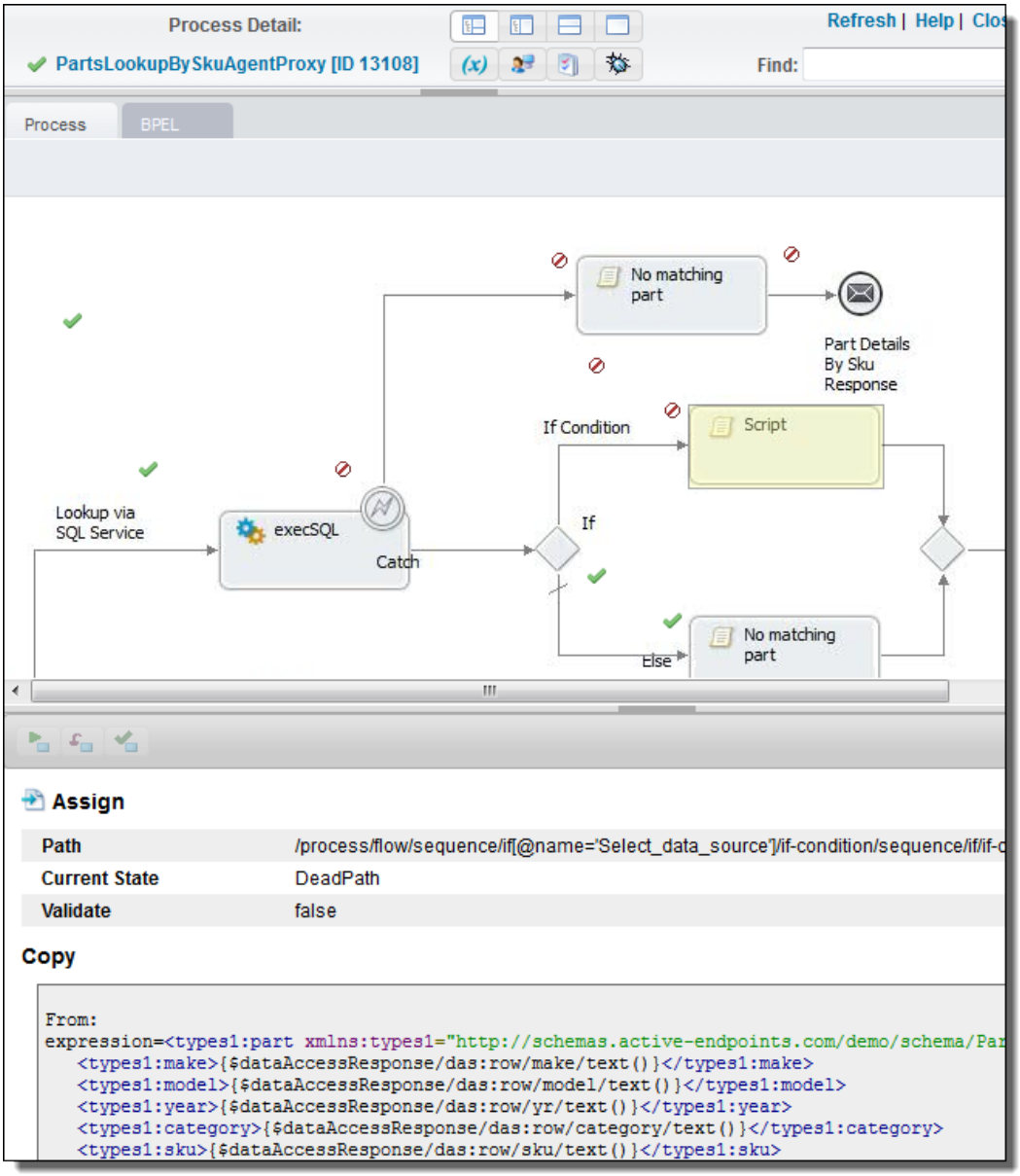
The Debug Threads dialog lets you suspend a process, step through it, and edit a variable, partner link, or correlation set.



The Canvas View

The Canvas view shows the main process flow and the execution path through the process. You can also view the process fault, event, and compensation handlers has them by selecting it from the Outline. For example, select a Catch to display all fault handlers.

The canvas view looks similar to the following example.



To view details, do the following:

- Select an activity from the diagram to view its properties.
- Select an activity from the diagram to put the activity in focus in the Outline, or vice versa.
- Select a People activity (if you are using Process Developer) to see details about active tasks.

To print the diagram, select a browser printing option from the right-mouse menu. The diagram prints with the same caption that appears in the canvas view. The timestamp indicates when the page was opened or refreshed.

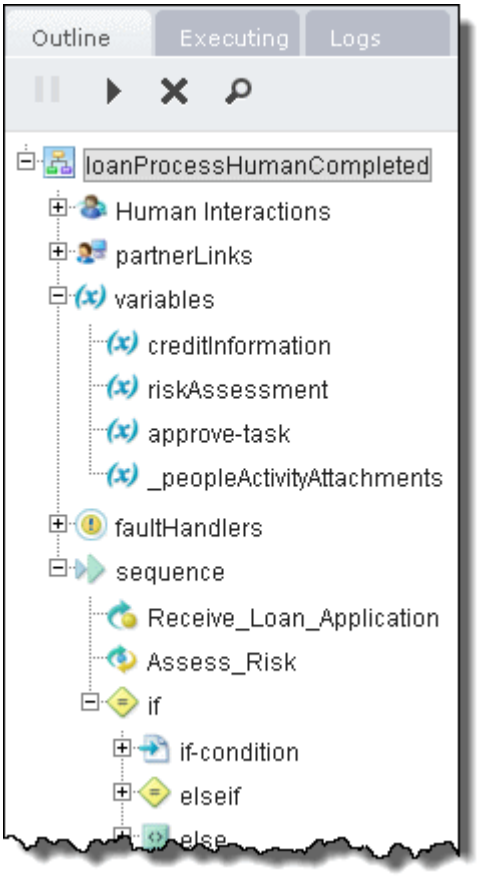
Executing View

The Executing view is a filter that shows only currently executing activities. This view is useful to locate the execution point to perform process exception management.

Outline View

Use the Outline view to select an activity so to put it in focus on the canvas. The activities are not necessarily shown in execution order: the order matches the order in which the activities were created. Also, some of the items on the outline are containers, such as a flow or a sequence and these may be hidden on the canvas. If you select a flow, the first activity within the flow is put in focus. If you select a collapsed container, shown with a plus sign, you will see a drill-down view of the activities within the container. If you select partner links or variables, you will see them in the Properties view.

The following illustration shows an Outline view of a process instance.



Outline View Menu Bar

The **Suspend**, **Resume**, and **Terminate** buttons are enabled only if the process is currently running or suspended. Using these features, you can perform process exception management by correcting, resuming, retrying or completing a suspended or suspended-faulting activity. .

Structural Elements of a BPEL Process

To view details about a process element, select it. The following table describes each element. Only elements that exist in the process are displayed here.

Process Element in Outline	Details Displayed in Properties
Activities	The activities section of the Outline begins with a flow (or other) activity that represents the main container for the whole process. Within the flow, there is a list of all process activities. The activities are in the same order as in the BPEL XML code. If the process was designed in Process Developer, the order matches the Outline view order. The activity list shown is not necessarily in execution order. For each activity, you can view the execution state and activity definition.
Correlation sets	These elements display the message properties (the definition and current value) that track different conversations carried on by the process. A correlation set contains a message property to ensure that each process conversation is uniquely identified.
Event handlers	Event handlers run concurrently with a process scope and invoke an activity based on an alarm or message received. The information displayed is name, state, and details of event handling activity. Scopes can have their own local fault handlers.
Fault handlers	Fault handlers run concurrently with a process scope and invoke an activity based on an alarm or message received. They display catch faults thrown by process activities. This includes name, state, and details of fault handling activity. Scopes can have their own local fault handlers.
Human Interactions	This is an Informatica extension for human tasks. It displays logical people groups, tasks, and notifications
Links	If an activity is the source of a link, the link is displayed below the activity node. Link properties are displayed, including link status (whether or not the link executed), the transition condition, if it exists, and the link's target activity.
Partner links	The types of partner links; that is the Web services that are invoked. It displays the partner role and/or my role. This may be the endpoint reference of the partner link service. You can see the address information in the Properties view. Scopes can have their own local partner links.
Process name	This is the local part of the process-qualified name. It shows the current state, start/end time of process instance, and deployment details for the process. Fault details may also be displayed.
Termination handlers	Activities can be terminated in a process through fault handling, early termination by a completion condition on a parallel `forEach` or through the execution of a Process Developer extension break activity.
Variables	These display the message or other data received, manipulated, and sent from the process. A variable may include attachments. For a variable type, you will see the message, schema type, or schema element. The current value of the variable is also displayed. For a running process, the value is current as of the time you opened or refreshed the Process Details window. If the variable has attachments, they are also displayed and can be downloaded.

Process States

The process can have the following execution states:

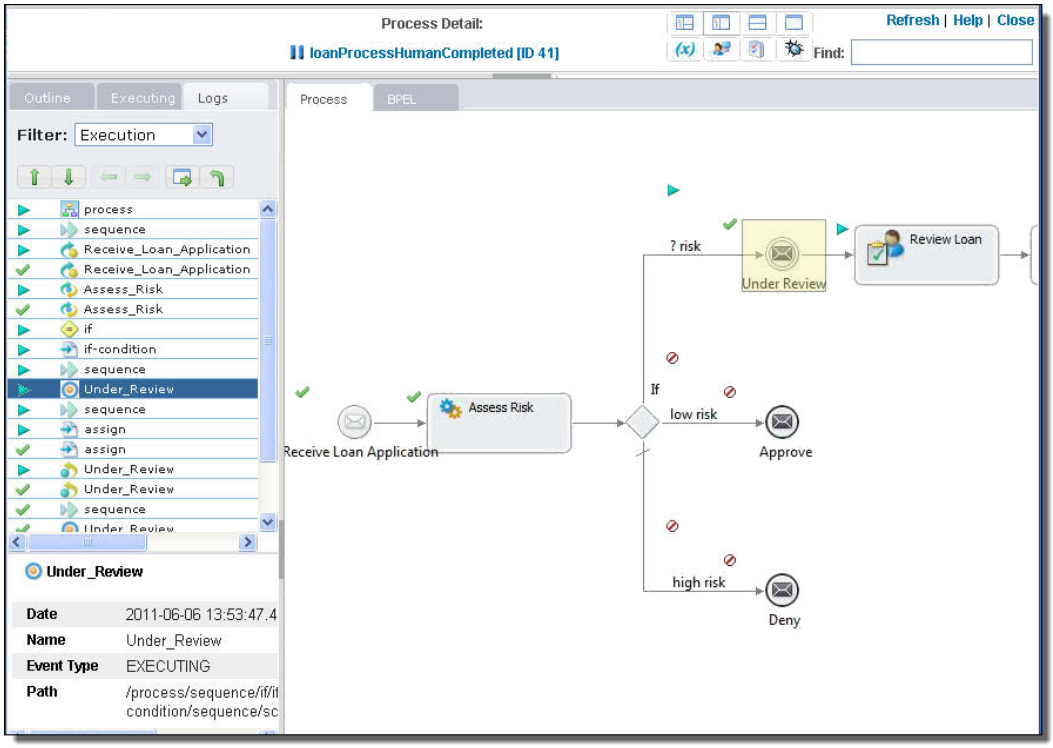
State	Meaning
Completed	Normal completion
Faulted	Completed with a fault or termination
Running	Snapshot of the executing process when you open the Process Details window. The process continues to run, but the Process Details window is not updated unless you select **Refresh**.
Suspended (Faulting)	Execution is stopped on a faulting activity. The activity has an uncaught fault and the process is configured for suspension on an uncaught fault.
Suspended (Manual)	The process stops running when you select **Suspend** from the Process Details window.

Additional states are shown when applicable for process migration warnings and errors.

Log View

In the Active Process Detail page, select the Logs tab to view a time-based list of executing activities. You can use the log to analyze process execution events, including data changes and faults. Navigate within the execution log to any point in time, for example, an iteration in While loop, and view the process state and data at that execution point. You can step through the events to see how a process changed over time.

The following shows an execution log.



Use the execution log as follows:

- Select a Filter to show data changes, faults, activity execution states, rewind points, or all. The Data Change filter shows variables whose content has changed, and also shows partner links and Logical People Groups that have been changed dynamically in an assign activity.
- Highlight an event and select Go To Execution Point to put the event in focus on the canvas. The process's state and data are based on a snapshot created at the selected log event.
- After selecting Go To Execution Point, return to the currently executing activity in a running process by selecting **Go To Current** in the upper-right corner of the window.
- Once you select an event, switch back to Outline view to see a graph of the process in the selected execution state. Also from the outline, you can inspect the variables at that point in time.

Activity States

You can determine the execution status of each activity by looking at the icon next to the activity.

Icon	Explanation
	Executing
	Ready to Execute
	Finished
	Faulted. Occurs when a fault is thrown during the execution of an activity.
	Suspended (Faulting). Occurs when a fault is thrown during the execution of an activity and the fault is not caught. If desired, you can make corrections or resume suspended (faulting) processes.
	Terminated. Occurs when the process is manually terminated.
	Dead Path
	Suspended
(none)	Inactive (the initial state of an activity)

For a running process, the icon next to an activity may change if you refresh the Process Details window.

Fault information includes:

Column	Explanation
Fault Name	BPEL or engine fault name
Fault Namespace	BPEL or engine fault namespace
Fault Source	Process activity that threw the fault
Fault Message Data	Data in the throw or catch fault variable
Fault Attachments	Content external to the variable may be included and can be downloads

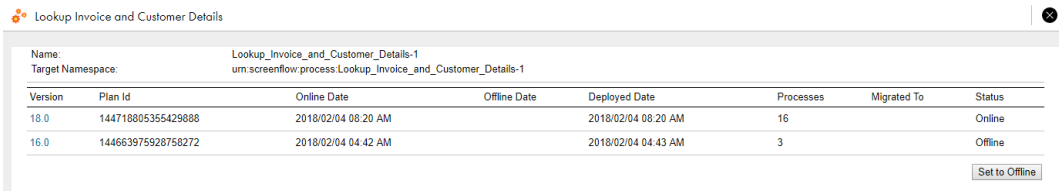
You can get further information about faults:

- Select the faulted activity to view the fault name.
- For a faulting activity, you can correct data, retry or complete the activity.

Tip: You can view the entire process log as a text file and download it by selecting **View Log** (the magnifier icon) on the Outline tab.

Process Version Listing Page

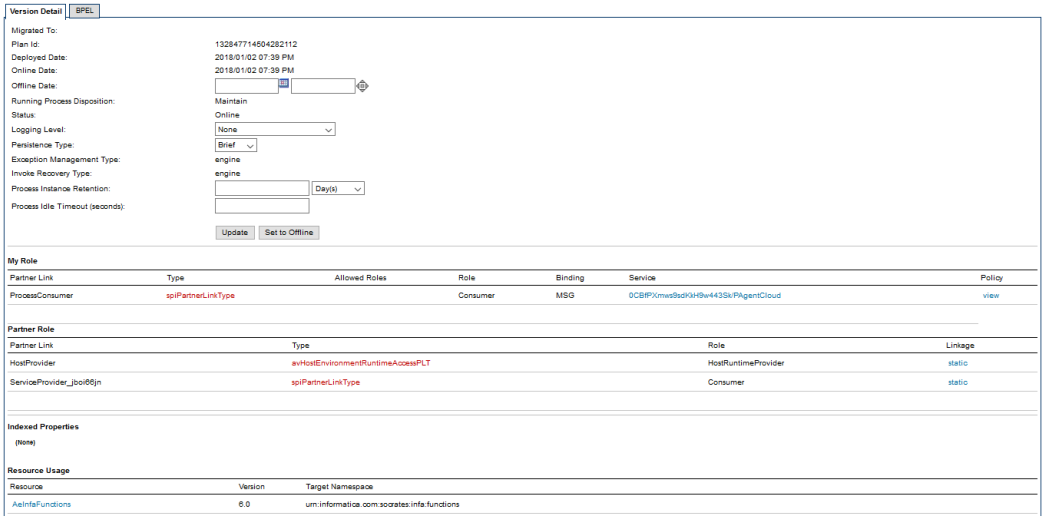
The following image shows a sample **Process Version Listing** page:



Property	Description
Version	The version number is automatically incremented when a new process version is deployed, unless a version number is specified in the deployment descriptor. The format of the number is `N.nn`, where `N` is major and `nn` is minor.
Plan ID	ID assigned to this version and associated disposition of running processes
Online Date	If an online (effective) date was not specified in the deployment descriptor, the online date is the same as the deployed date. A process is online immediately when deployed, unless an online date is specified.
Offline Date	A process version does not have an offline (expiration) date unless one is specified in the deployment descriptor. By default, a process version automatically goes offline when a newer version comes online.
Deployed Date	Date the process is added to the engine database
Processes	Number of active process instances
Migrated To	Process version that this version migrates to
Status	Online, Online Pending, Offline, Offline Pending

Deployed Process Version Detail Page

The **Deployed Process Version Detail** page displays all the details from the process deployment descriptor as well as the process definition.



The sections of this page are described in the following topics:

- Process Version Life Cycles
- Updating a Process Version
- Setting a Process Version Offline or Online
- Logging Level
- Process Version Persistence Type
- Exception Management Type
- Invoke Recovery Type
- Process Instance Retention
- Deployed Process Detail Graph

The Deployed Process Version Detail page also shows endpoint reference details and other related details:

Role	Explanation
My Role	My Role partner link endpoint reference details are generated from information in the deployment descriptor. Rest your mouse on the partner link type to view the associated namespace. Select the Service Name link to view the WSDL file for the Web Service exposed by the partner link. Select View in the Policy column to view any attached policy assertions.
Partner Role	Partner Role partner link endpoint reference details are generated from information in the deployment descriptor. Rest your mouse on the partner link type to view the associated namespace. Select a static endpoint type from the Linkage column to view the endpoint definition.
Indexed Properties	Indexed Properties, if any, are displayed.
Event Filters	Event Filters, if any, are displayed. An event filter specifies process events that are passed to the Event Manager for processing. Event services, deployed to the Catalog, act upon the events.
Resource Usage	Resource Usage shows WSDL, schema, and other files and their target namespace referenced in this process. Select a Resource to view the resource definition.

Understanding Process Version Life Cycles

Process versioning allows different versions a process to exist in Process Server. Process versioning allows you to control when processes become effective and for how long. You can also control what happens to processes created by older versions when a new version becomes effective. While multiple versions of a process can exist concurrently, only the latest effective version can create new process instances.

The latest effective version is in an online state. Other states are:

- **Online pending**: describes versions that have an effective date in the future.
- **Offline pending**: describes versions whose expiration date has arrived, but running process instances are still active.
- **Offline**: describes expired versions that no longer have running process instances.

The process deployment descriptor (PDD) can contain a version element whose attributes describe how a deployment is versioned. These selections are all optional and have default values as described below.

The following example shows the syntax for version information in the `.pdd` file.

```
<version effectiveDate="2005-12-12T00:00:00-05:00"
         expirationDate="2007-12-12T00:00:00-05:00"
         id="1.5"
         runningProcessDisposition="migrate"/>
```

where:

- `effectiveDate` is the date the new version becomes the current version and all new process instances run against it.
  Depending on the disposition selected for running processes, some may continue to run until they finish using the older version. The effective date is an XML schema date/time value. The time expression includes a time zone, indicated as the midnight hour plus or minus the number of hours ahead of or behind Coordinated Universal Time (UTC) for the computer's time zone. In the example above, the computer time zone is Eastern Standard Time, which is five hours behind UTC. If you do not provide an effective date, it defaults to the date and time the process is deployed to the server.
- `expirationDate` is the date, beyond the online date, the current version expires.
  An offline version is not capable of creating new process instances. Once all of the running processes tied to an offline pending version complete, the version becomes offline. All process instances for the current version run to completion. The expiration date is an XML schema date/time value. (Same as effective date). If you do not provide an expiration date, the version does not expire until you manually set it to offline in the Application Integration Console or until a newer version is deployed.
- `id` is the process version number in `major.minor` format.
  You do not need to provide a version number as Process Server auto-increments new versions. The server increments a version number by dropping the minor value and adding 1 to the max number. For example, version 1.5 increments to version 2.0.
- `runningProcessDisposition` is the action Process Server takes on any other versions of the same process that currently have processes executing after this version's effective date arrives.
  Values for `runningProcessDisposition` are:
  - `Maintain`. All process instances for the previous versions should run to completion. This is the default value.
  - `Migrate`. All running process instances created by previous versions will have their state information migrated to use the newly deployed process definition once its effective date arrives. If there are incompatible changes between the versions, descriptive warning messages are written to the Application Integration Console Server Log. Refer to the *Migrating Running Processes when Warnings are Generated* for details.

- `Terminate`. Indicates that all process instances running under previous versions should terminate on the effective date of the new version, regardless of if the process instances are complete.

Updating a Process Version

You can view this property on the **Deployed Process Version Detail** page.

The options for updating a process version are:

- Depending on the version status (Online, Online Pending, Offline, Offline Pending), you may be able to update the online date, offline date, and running process disposition. For example, you can add an Offline Date to the online version, and select **Update**.
- You can also set the Process Instance Retention Days and update the process version. For details, see *Process Instance Retention*.

Setting a Process Version Offline or Online

You can view this property on the "Deployed Process Version Detail Page" on page 89 page.

The options for setting a process version offline or online are:

- To inactivate this version at a specified date and time, type in the values in the Offline Date fields.
- To inactivate this version immediately, select **Set to Offline**. No new process instances can be instantiated from this version. The status is Offline Pending until all running processes complete.
- If you had previously set this version offline, you may be able to restore it. To restore this version to the online or online pending (future) version, select the available option, **Restore to Online** or **Restore to Online Pending**.

Setting the Logging Level

You can set this property on the **Deployed Process Version Detail** page.

You can view or download an execution log for a running or completed process. An execution log provides start and end times for activity execution and helps you troubleshoot faulted processes. The following list describes the different logging levels:

- **None:** The Process Server does not log any information. Use this option to enhance engine performance.
- **Fault:** The Process Server logs only fault information. If no faults occur in a process, the Process Logging level defaults to **None**. Select **Fault** to reduce the size of the log file.
- **Execution:** This is the default option. The Process Server logs all execution statements except for *Will Not Execute* statements. Select **Execution** to decrease the size of the log file.
- **Execution with Service Data:** The Process Server logs all execution and fault information, as well as some WSIO activity information. For execution information, Process Server logs deadpath states, terminations, ready-to-execute, and so on. For WSIO, Process Server logs invokes, picks, and receives, but excludes information related to data assignment or changes.
- **Execution with Data:** The Process Server logs all execution statements except for *Will Not Execute* statements, but includes variable, expression, and partner link data. Select **Execution with Data** to decrease the size of the log file.
- **Full**: The Process Server logs all execution statements *Will Not Execute* statements for deadpath activities. For example, the Process Server logs all fault handling statements that are not executed.
- **System Default:** When you use this option, the logging level for the process version corresponds to the process logging level property that is set on the corresponding Process Server.

To improve processing speed, perform no logging or minimum logging. Informatica recommends that you use the **None** or **Terse** logging levels.

Setting the Process Version Persistence Type

You can set this property on the "Deployed Process Version Detail Page" on page 89 page.

Persistence refers to storage of active processes. By default, when a process runs on the server, all state and variable data is stored in the Process Server database. However, this setting can be changed in the PDD file to increase server performance and reduce database size.

Persistence setting selections are as follows:

Setting	Explanation
Full (default)	For each process instance, all running, faulted, and completed state information is stored. In the event of a server failure, a running process can be fully recovered. The recovery is possible because this setting tells Process Server to maintain a journal (a record of the changes intended for the database). **Note**: If the process uses a WS-RM invoke handler for a partner role or a WS-Reliable Messaging policy assertion on a my role, full persistence is required.
Persist	Same storage setting as Full, but without journaling. If processes are running and the server fails, processes are suspended. The process is recoverable if the system goes down, but needs to be looked at since no journaling was done. This process is marked as suspended.
Final	Stores only the final state of the process (completed or faulted) and process variables. On a server failure, a running process is terminated. This setting makes fewer database writes than the previous two settings, but still allows you to view a graph of the process on the Active Processes Detail page in the Application Integration Console. Here, you can see the execution path and final values of process variables. A process runs only in memory, and the Server Property called Process Idle Timeout has no effect on this persistence level.
Brief	This is the minimum level for process logging. Process Server stores only the start and completion times and the final state (completed or faulted). Also, it stores state and process variables only if the process faults. A process runs only in memory, and the Server Property called Process Idle Timeout has no effect on this persistence level.
None	No process information is stored in the server database when a process terminates. The process instance is not listed in the Processes page.

Invoke Recovery Type

You can view this property on the "Deployed Process Version Detail Page" on page 89 page.

For invoke activities that do not complete because of a node failure, you can suspend the process upon recovery. The process is suspended at the pending invoke, and you can perform process exception management.

An individual process can override the engine setting with an entry in the Process Deployment Descriptor (PDD) file. The settings are:

Setting	Explanation
System Default	The current engine setting for all processes. The default engine setting is to disable invoke recovery; however, the current setting may be different.
False	Do not allow a pending invoke activity to suspend upon recovery. The process will terminate abnormally. This setting overrides the engine setting.
True	Suspend a pending invoke on recovery to put it in a suspended-faulting state. You can then perform process exception management on the invoke activity, followed by retrying or completing the faulting activity or scope. This setting overrides the engine setting.

Process Instance Retention

You can view the **Process Instance Retention** property on the Deployed Process Version detail page.

You can specify how long to keep completed and faulted processes in the Process Server database before deleting them on an automated schedule. This setting is available in the following locations:

- Process Deployment Descriptor editor in Process Developer. You can add the setting to a process's deployment descriptor.
- The Deployed Process Version detail page of the Application Integration Console. Add or change the setting for a deployed process.
- The default retention days setting for all processes on the Application Integration Console. The setting for an individual process overrides this setting.

The retention setting applies to a process version with a status of online or online pending, and not to offline pending or offline versions. The schedule begins on the completed date or faulted date of each process, as shown on the **Processes** page. For example, if a process instance completes on December 31, and the retention setting is 30 days, the process instance is deleted from the database on the next scheduled deletion after January 30.

Configuring Retention for an Individual Process

1. In the Application Integration Console, select **Deployed Assets** > **Deployed Processes**.
2. Select a process from the list.
3. If multiple versions exist, from the **Deployed Process Detail** page, select an online or online pending version to open the **Deployed Process Version Detail** page.
4. In the **Process Instance Retention** field, specify the number of retention days, hours, or minutes for the process version.
5. Select **Update**. The change takes effect immediately for all completed and faulted processes.

For details about scheduled database maintenance, see *Storage*.

Deployed Process Detail Graph

The Deployed Process Detail Graph page presents many details about a process instance:

- An **Outline** view shows the structural elements of a BPEL process. You can select an element to view its properties and values.

- A **Graphic** view shows the main process flow. If the process has event handlers, fault handlers, and compensation handlers, you can view them by selecting a tab. You can also select an activity to view its properties.
- A **Properties** view appears for a selected element. For example, when the Process element is selected from the Outline view, you can see process properties and their current values.

Process Restart

Users who have the Admin or Operator role can restart a completed, faulted, or suspended process from the Application Integration Console. The **My Processes** tab in Application Integration lists all the processes that you had run with authentication. You can restart a completed, faulted, or suspended process from the **My Processes** tab.

You can restart a process that meets the following criteria:

- The process was run at least once on the Cloud Server or on an agent.
- The process is in one of the following states:
  - Completed
  - Faulted
  - Suspended
- The tracing level for the process is set to **Normal** or **Verbose**.
- If the process contains attachments, the persistence type is set to **Full** or **Persist**.

You can also restart a process that has the **Binding** property set to **Event**.

You cannot restart a BPEL process that has the **Create Instance** property to **Yes** in multiple steps.

If you edited a process but did not run it, when you restart the process, the last run version of the process is restarted.

if you restart a process that contains a subprocess, the subprocess is also restarted. If a parent process contains multiple subprocesses, you can choose to restart specific subprocesses.

Restarting a Process from the Application Integration Console

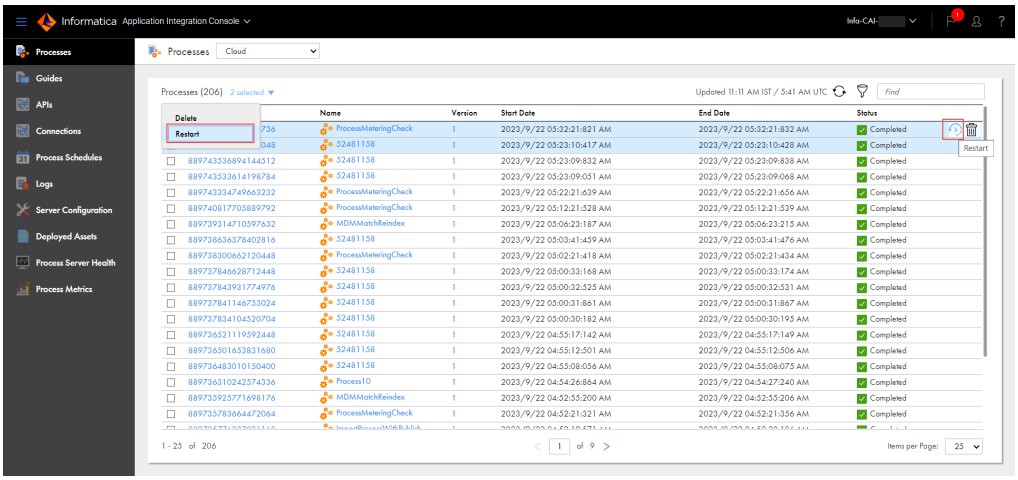
You can restart a completed, faulted, or suspended process from the Application Integration Console.

1. In Application Integration Console, click **Processes** on the left navigation pane.
2. Select the process that you want to restart, and then click the **Restart** icon in the row that contains the process instance.

   To restart multiple processes simultaneously, select the processes and perform one of the following steps:

   - Select **Restart** from the **Selected** list.
   - Right-click any selected row and select **Restart**.

   The following image shows the **Restart** options available on the **Processes** page:



The selected processes are restarted and listed with different run IDs. You can click a run ID to view details about the process execution.

Restarting a Process from Application Integration

The **My Processes** tab in Application Integration lists all the processes that you had run with authentication. You can restart a completed, faulted, or suspended process from the **My Processes** tab.

The following video shows you how to restart a process from Application Integration:

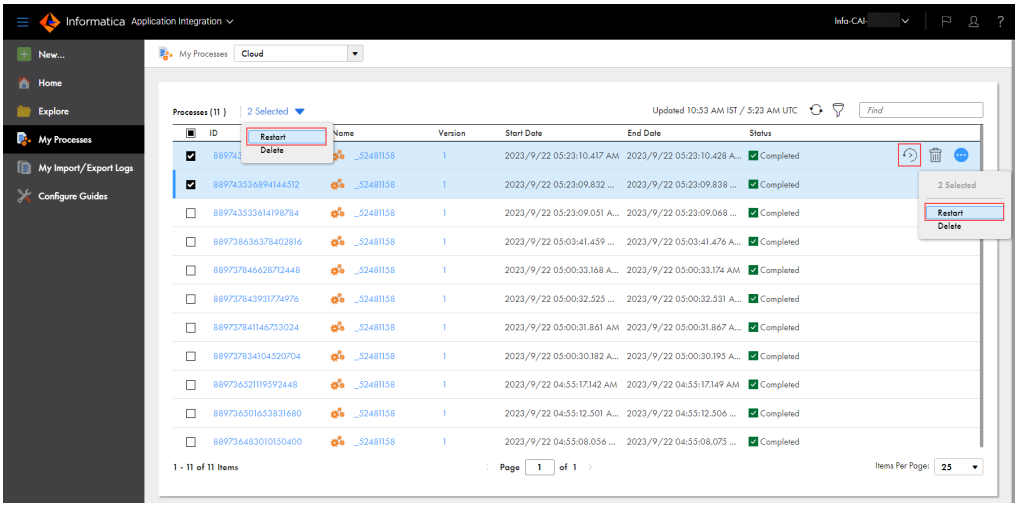
https://knowledge.informatica.com/s/article/DOC-18507

1. In Application Integration, click **My Processes** on the left navigation pane.
2. Select the process that you want to restart, and then click the **Restart** icon in the row that contains the process instance.

   To restart multiple processes simultaneously, select the processes and perform one of the following steps:

   - Select **Restart** from the **Actions** menu.
   - Select **Restart** from the **Selected** list.
   - Right-click any selected row and select **Restart**.

   The following image shows the **Restart** options available on the **My Processes** page:



The selected processes are restarted and listed with different run IDs. You can click a run ID to view details about the process execution.

CHAPTER 6

Process Server Health

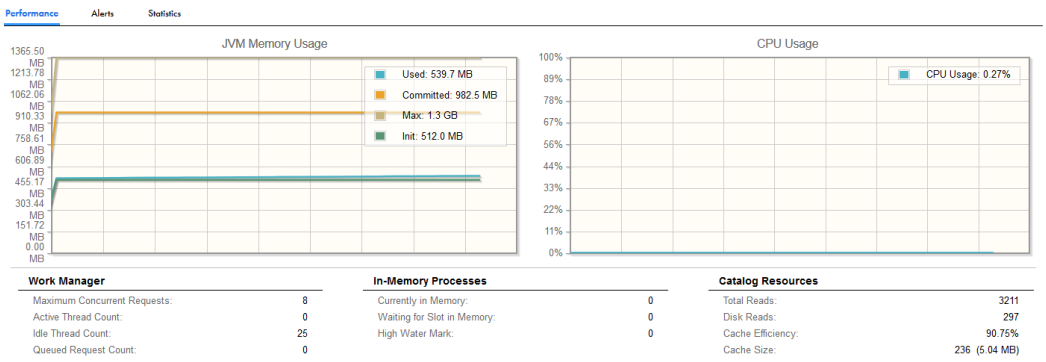
You must select a Secure Agent to see Process Server health details.

Performance

You see these details on the Performance tab:

- Overall performance data.
- Performance and throughput of service requests.
- Resource and service bottlenecks.

For example:



The statistics show a snapshot of what was in memory when you first open the page. Refresh the page to get updates.

These statistics let you ask questions such as:

- How many requests are being sent to the server?
- How long is the queue?
- How many requests are waiting?
- How long does it take to provide a response?
- How long is it taking for the server to return from service calls?
- Should I increase the work manager limits set on the Server Properties page?

After you run a load test, you can watch the metrics and set or reset configuration properties.

Platform Support

As you analyze performance in relation to your application server, keep the following in mind:

- Statistics important to consider before changing other settings include:
  - CPU Utilization—graph over time interval
  - JVM Memory Usage—graph over time interval
  - Database Connection Pool—available, in use, waiters, and high water marks

Alerts

You see alerts triggered by server statistics monitoring thresholds.

During each threshold interval, if an error or warning occurs, it is reported at the end of an evaluation period. For example, if your threshold interval is set to five minutes, and the evaluation frequency is set to five times per interval, the first trouble item may appear after one minute has elapsed, but will not be reported more than once per threshold interval. If competing threshold levels occur during same evaluation cycle, the highest severity is reported.

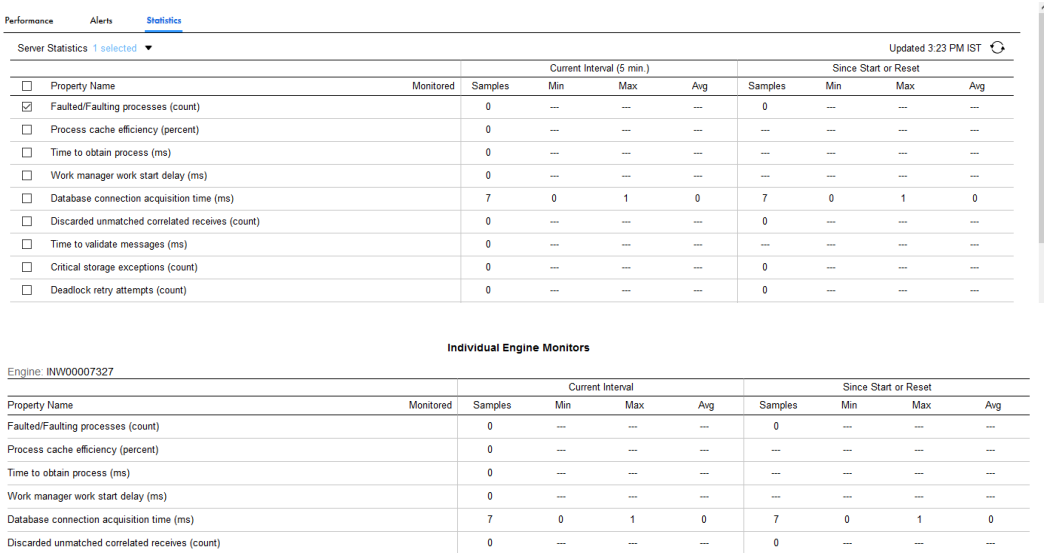
Each time you refresh your browser or open the Monitoring page, monitoring statistics are updated. The report, however, is updated only at the end of an evaluation period.

The list of trouble items shows the error/warning level, time, property name, and configuration details for each item reported.

Statistics

Note: You only see this command if you have selected an agent.

Use the Statistics page to view the monitoring properties you configured or which are the default. Process Server collects engine statistics and then aggregates them by intervals. If desired, you can configure a threshold interval and an error/warning level for each of these properties. If you configure these settings, you will see additional signals displayed on this page, such as an eyeglass icon indicating a property is being watched.



Performance Alerts Statistics

Server Statistics 1 selected

Updated 3:23 PM IST

	Property Name	Monitored	Current Interval (5 min.) Samples	Min	Max	Avg	Since Start or Reset Samples	Min	Max	Avg
☑	Faulted/Faulting processes (count)		0	---	---	---	0	---	---	---
☐	Process cache efficiency (percent)		0	---	---	---	---	---	---	---
☐	Time to obtain process (ms)		0	---	---	---	---	---	---	---
☐	Work manager work start delay (ms)		0	---	---	---	---	---	---	---
☐	Database connection acquisition time (ms)		7	0	1	0	7	0	1	0
☐	Discarded unmatched correlated receives (count)		0	---	---	---	0	---	---	---
☐	Time to validate messages (ms)		0	---	---	---	---	---	---	---
☐	Critical storage exceptions (count)		0	---	---	---	0	---	---	---
☐	Deadlock retry attempts (count)		0	---	---	---	0	---	---	---

Individual Engine Monitors

Engine: INW00007327

Property Name	Monitored	Current Interval Samples	Min	Max	Avg	Since Start or Reset Samples	Min	Max	Avg
Faulted/Faulting processes (count)		0	---	---	---	0	---	---	---
Process cache efficiency (percent)		0	---	---	---	---	---	---	---
Time to obtain process (ms)		0	---	---	---	---	---	---	---
Work manager work start delay (ms)		0	---	---	---	---	---	---	---
Database connection acquisition time (ms)		7	0	1	0	7	0	1	0
Discarded unmatched correlated receives (count)		0	---	---	---	0	---	---	---

During a collection interval, Process Server maintains statistics for each configured property value, including the maximum, average, and total count values. In addition, historical statistics are collected. Historical statistics include minimum and maximum values for the intervals being recorded. Statistics are collected in-memory.

Select **Reset** to clear statistics. For example, if you correct an error condition, you can reset the statistics collection starting with zero values to see if performance improves. Once all errors and warnings are cleared, the engine monitoring level is reset to Normal.

CHAPTER 7

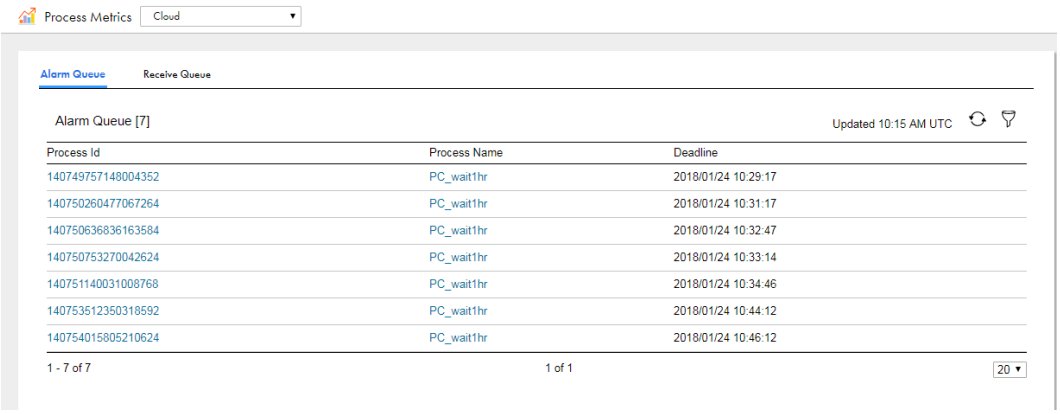
Process Metrics

Filter and view metrics for alarm queues and receive queues.

Alarm Queue Metrics

The **Alarm Queue** page lists the alarms in the alert queue, organized by process.

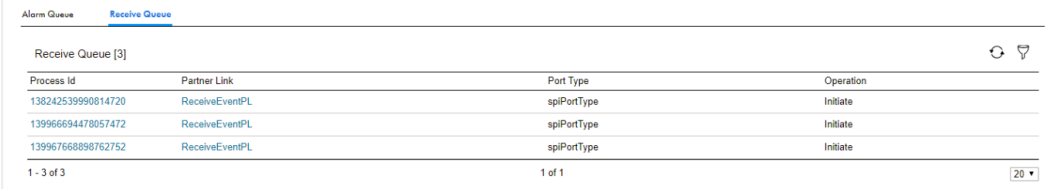
The following image shows a sample **Alarm Queue** page:



Use the selection filters to determine which process activities appear when you view active alarms:

Option	Description
Deadline Between	Beginning and Ending date and time for alarm.
Process ID	Process instance ID. You can find this ID on the Processes page.
Process Name	Local part of the process-qualified name (`qname`) .
Group	The Group this process belongs to (optional). The Group name displays in the On Alarm list.
Hide System	Uncheck to display the default System processes. The System processes are hidden by default.

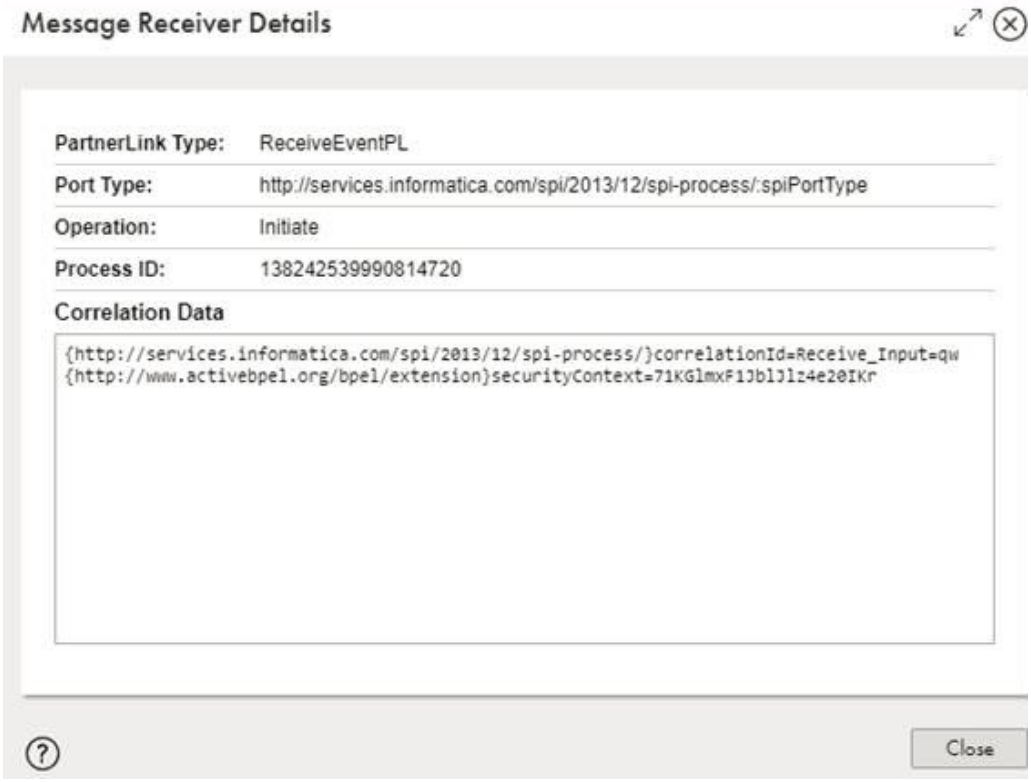
Receive Queue Metrics



Select one or more options from the filter option list to view a selection of active receives

If you click on:

- **Process ID**: the **Process View Detail** page displays.
- **Partner Link**: **Message Receiver Details** window appears:



You see the following information on the **Message Receiver Details** window:

Item	Meaning
Process ID	Process Instance ID. You find this ID on the Active Processes page.
Partner Link	The partner link for the item in the receive queue. A partner link is a communication exchange between two partners. In the most basic form, the process is a partner link of an external service, receiving a request from it. A partner link defines the role that the process plays (if any) and the role that the partner service plays (if any) in the particular exchange

Item	Meaning
Port Type	The port type, which is a port type in WSDL is a set of related operations such as receive, reply, and invoke include operations.
Operation	The operation that just executed.

CHAPTER 8

APIs

The APIs page shows a list of **APIs** of published processes that use the REST/SOAP binding, along with the REST and SOAP endpoints and their status. Use the **APIs** page to activate or deactivate the APIs of published processes.

You can activate or deactivate the APIs, that is, the REST and SOAP endpoints from the **APIs** page. With this option, you don't have to unpublish and republish a process to deactivate and activate the API. When you publish a process from Application Integration for the first time, the API is activated and displayed on the **APIs** page in Application Integration Console. Later, when you activate or deactivate the API from the **APIs** page, Application Integration also updates the API status in the **Properties Detail** dialog box of the published process. Republishing a process with an inactive API status does not activate the API. You must activate it from the **APIs** page. If you unpublish the process from Application Integration, the API is removed from the **APIs** page.

Note: Activation or deactivation does not apply to scheduled processes and subprocesses. Activation or deactivation of a scheduled process API does not affect schedules. If a process contains a subprocess, and you deactivate the API of the parent process, the subprocess inside it still runs.

Example

Consider a bank that offers fund transfers on its web portal through APIs developed using Application Integration. The bank uses an Oracle database to store transaction information for future audits and monitoring.

During database maintenance, the bank wants to temporarily stop transactions to avoid potential issues with incoming requests.

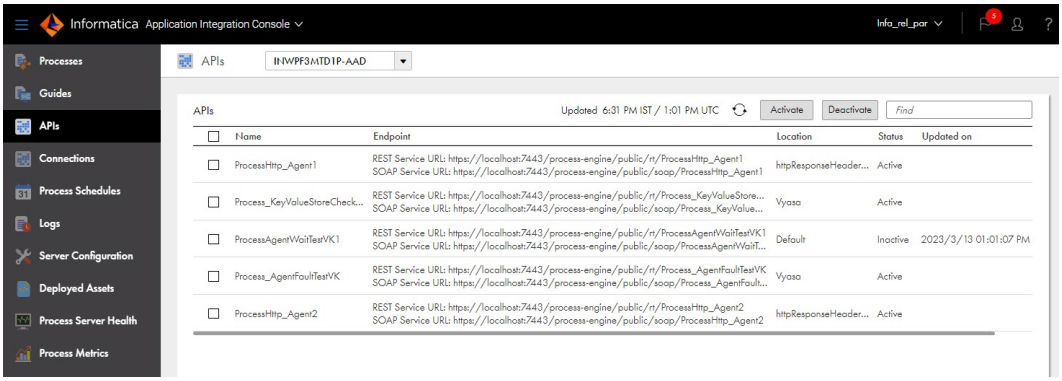
In this case, the bank can filter and find all APIs that enable fund transfers and deactivate them. If customers invoke the deactivated APIs during the downtime, they will see a `503 Service Unavailable` or an equivalent message. After maintenance is complete, the APIs can be activated to resume fund transfers.

Viewing APIs

To view the APIs of published processes, in Application Integration Console, click **APIs**. By default, the **APIs** page lists the APIs for processes published on the Cloud Server.

You can select the Cloud Server or a Secure Agent, and enter the API name of the process to search for an API.

The following image shows the **APIs** page:



The **APIs** page displays the following information:

Name

Displays the names of the published processes on the Cloud Server or the selected Secure Agent.

Endpoint

Displays the REST and SOAP endpoint URLs.

Location

Displays the location of the project or folder where the process is saved.

Status

Displays the status of the API. The status can use one of the following values:

- Active. The API is activated and available for use.
- Inactive. The API is deactivated. If users invoke the API, a `503 Service Unavailable` error is displayed.

Updated on

Displays the last updated date and time of the API status.

Deactivating an API

If a user uses direct endpoints to run an API, and you want to stop the API usage for a certain period, you can deactivate the API. You can deactivate the APIs, that is, the REST and SOAP endpoints of published processes. If users invoke the deactivated API, a `503 Service Unavailable` error is displayed.

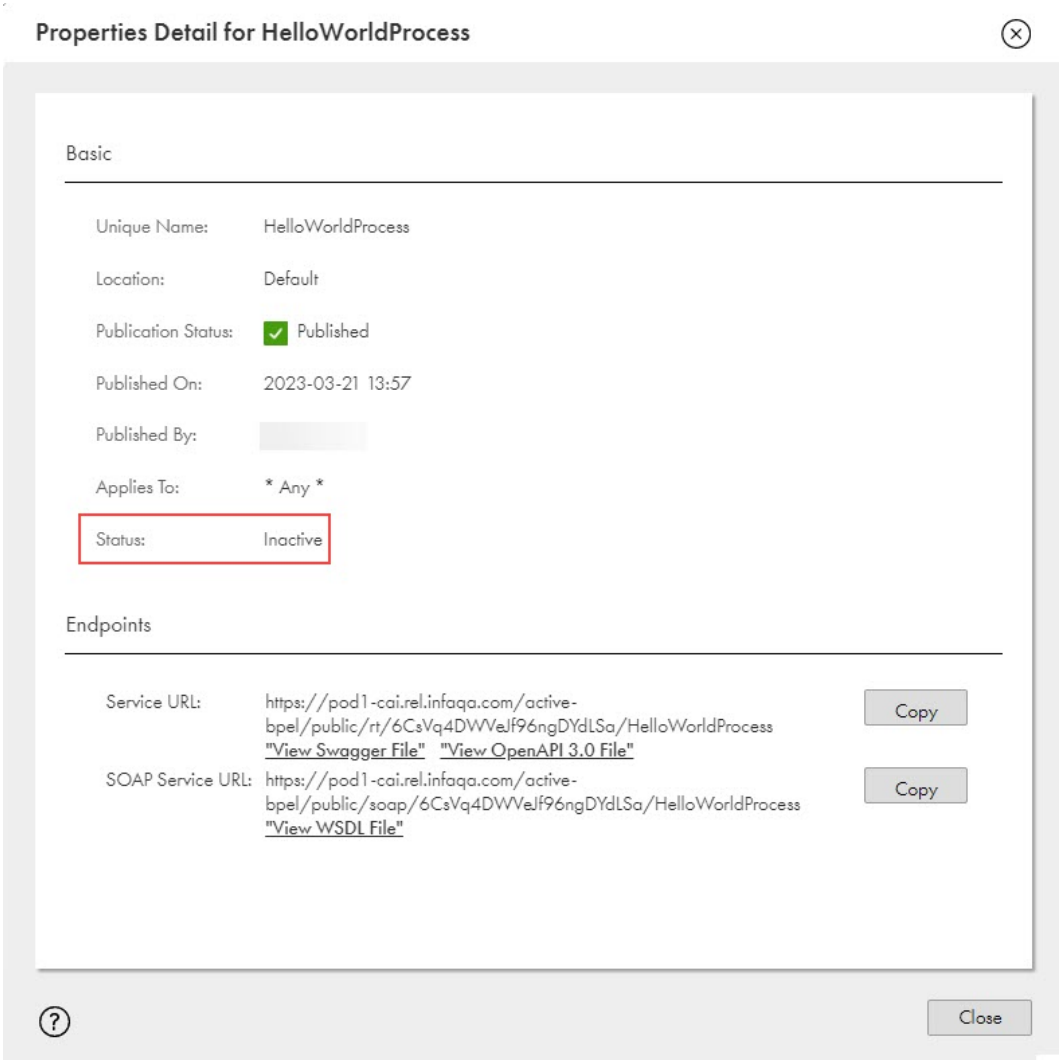
To deactivate the APIs, perform the following steps:

1. In Application Integration Console, click **APIs** on the left navigation bar.
2. Select the Cloud Server or a Secure Agent to see the APIs.
3. Select one or more APIs that you want to deactivate, and click **Deactivate**.
   The **Update API Status** dialog box appears.
4. To deactivate an API on the Cloud Server, click **Deactivate** to proceed.
   The status of the API is updated to **Inactive** on the **APIs** page.
5. To deactivate an API on the Secure Agent, perform one of the following steps:
   - If the process runs only on the selected Secure Agent, click **Deactivate** to proceed.

- If the process runs on more than one Secure Agent in a group, perform one of the following steps:
  - Click **Deactivate only on the selected Secure Agent** to proceed with the selected Secure Agent.
  - Click **Deactivate on all the Secure Agents in the group** to proceed with the Secure Agent group.

The status of the API is updated to **Inactive** on the **APIs** page.

Application Integration also updates the API status on the **Properties Detail** dialog box of the published process as shown in the following image:



If the status of the API appears as **Inactive** on the **Properties Detail** dialog box, you must activate the API to make the API available for the user.

Activating an API

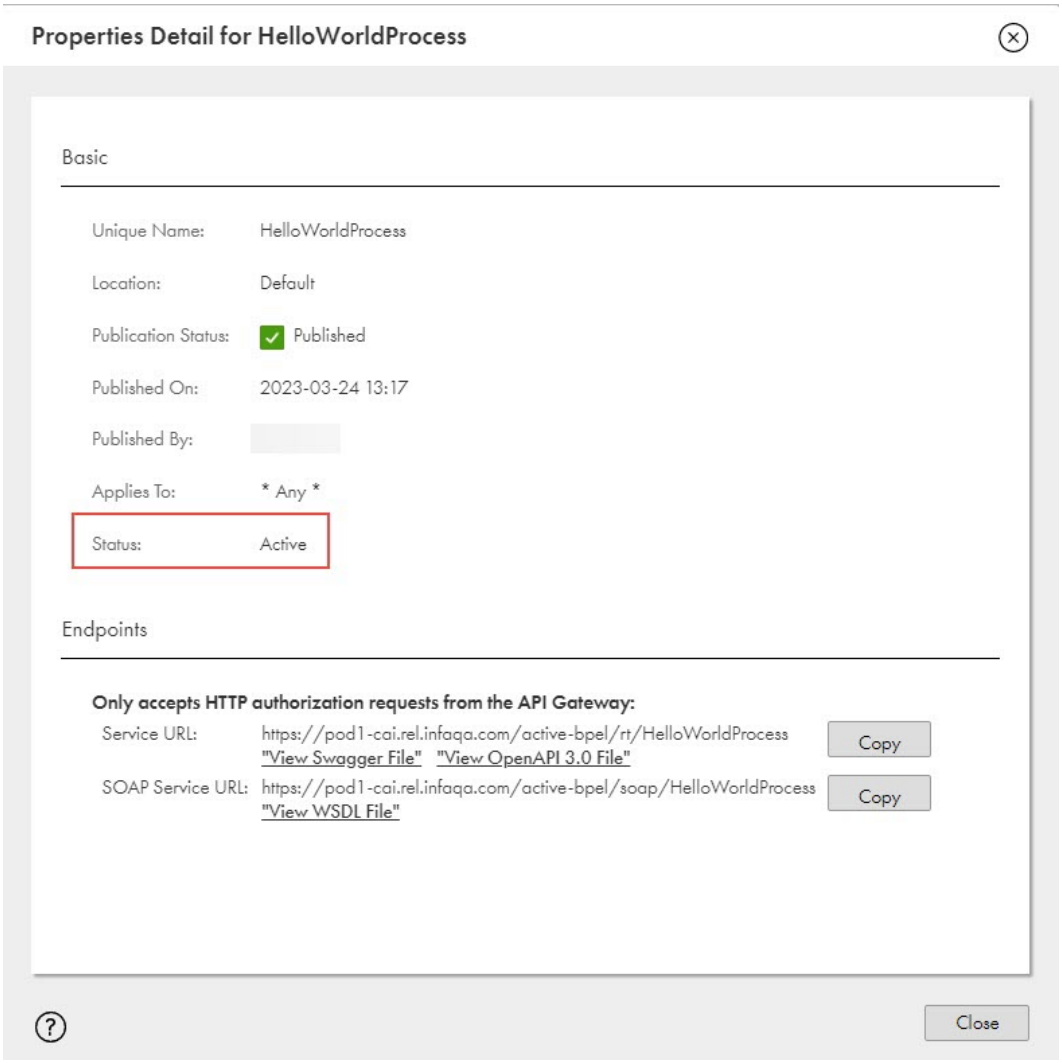
You can resume the usage of deactivated APIs by activating the APIs, that is, the REST and SOAP endpoints of published processes.

To activate the APIs, perform the following steps:

1. In Application Integration Console, click **APIs** on the left navigation bar.
2. Select the Cloud Server or a Secure Agent to see the APIs.
3. Select one or more APIs that you want to activate, and click **Activate**.
   The **Update API Status** dialog box appears.
4. To activate an API on the Cloud Server, click **Activate** to proceed.
   The status of the API is updated to **Active** on the **APIs** page.
5. To activate an API on the Secure Agent, perform one of the following steps:
   - If the process runs only on the selected Secure Agent, click **Activate** to proceed.
   - If the process runs on more than one Secure Agent in a group, perform one of the following steps:
     - Click **Activate only on the selected Secure Agent** to proceed with the selected Secure Agent.
     - Click **Activate on all the Secure Agents in the group** to proceed with the Secure Agent group.

   The status of the API is updated to **Active** on the **APIs** page.

Application Integration also updates the API status on the **Properties Detail** dialog box of the published process as shown in the following image:


Properties Detail for HelloWorldProcess
Basic
Unique Name: HelloWorldProcess
Location: Default
Publication Status: Published
Published On: 2023-03-24 13:17
Published By:
Applies To: * Any *
Status: Active
Endpoints
Only accepts HTTP authorization requests from the API Gateway:
Service URL: https://pod1-cai.rel.infaqa.com/active-bpel/rt/HelloWorldProcess
"View Swagger File" "View OpenAPI 3.0 File"
Copy
SOAP Service URL: https://pod1-cai.rel.infaqa.com/active-bpel/soap/HelloWorldProcess
"View WSDL File"
Copy
Close

CHAPTER 9

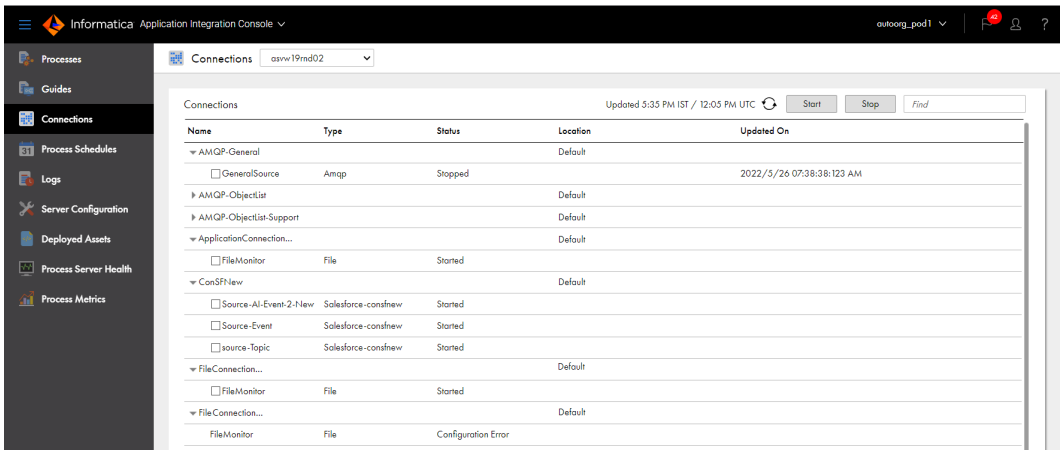
Connections

The **Connections** page shows a list of event sources in listener-based connections that have been published on a selected Secure Agent, along with the connection type and status. You can also view a list of event sources in Kafka connections that have been published on the Cloud Server. Use the **Connections** page to view, start, or stop event sources in listener-based connections.

Note: You can publish Kafka connections only on the AWS PODs on the Cloud Server.

You can enter a connection name or connection ID to search for a connection.

The following image shows the **Connections** page:



The **Connections** page displays the following information:

Name

Displays the listener-based connections published on the Cloud Server or a selected Secure Agent. Expand the connection to see the available event sources.

Type

Displays the type of the event source.

Status

Displays the status of the event source. The status can use one of the following values:

- Started. The event source is started.
- Stopped. The event source is stopped.
- Configuration Error. The connection was published with incorrect values. For example, if the IP address contains an alphanumeric character, Application Integration cannot connect to the endpoints. In this case, you must correct the configuration values and republish the connection.

Location

Displays the location of the project or folder where the connection is saved.

Updated On

Displays the last updated date and time of the event source.

You can click the **Download Log** icon to download the listener logs for the Kafka connection published on the Cloud Server. The log contains warnings and errors for the last 24 hours.

Starting and stopping an event source in a listener-based connection

You can start and stop event sources in listener-based connections that run on a Secure Agent or a Secure Agent group. You can also start and stop event sources in Kafka connections that have been published on the Cloud Server. With this option, you do not have to unpublish and republish the connection to stop and start the event-based processes.

To start and stop an event source in a connection, perform the following steps:

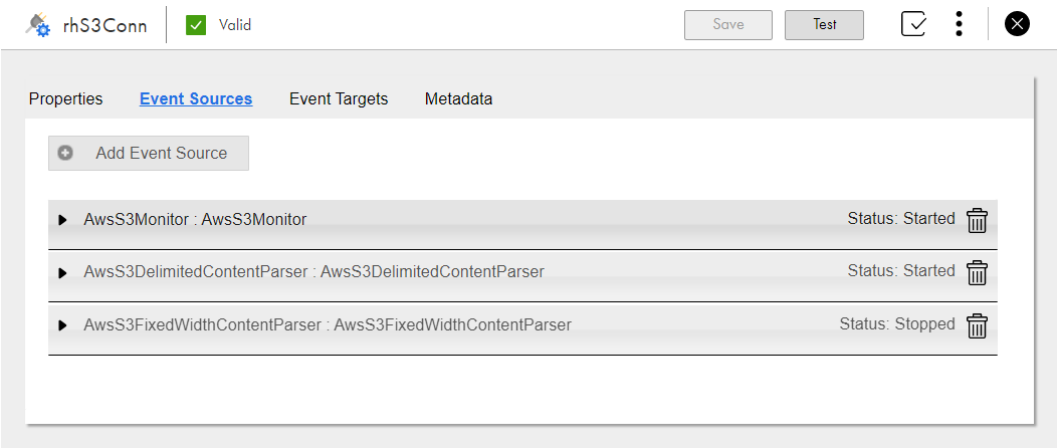
1. In Application Integration Console, click **Connections** on the left navigation bar.
2. Select **Cloud** or a Secure Agent to see the connections.
3. Expand the connections to view the event sources.
4. Select one or more event sources that you want to start or stop, and click **Start** or **Stop** accordingly.
   The **Update Listener Status** dialog box appears.
5. To start an event source in a connection that has been published on the Cloud Server, click **Start** to proceed.
   The status of the event source is updated to **Started** on the **Connections** page.
6. To stop an event source in a connection that has been published on the Cloud Server, click **Stop** to proceed.
   The status of the event source is updated to **Stopped** on the **Connections** page.
7. To start an event source in a connection that has been published on a Secure Agent, perform one of the following steps:
   - If the connection runs only on the selected Secure Agent, click **Start on the selected Secure Agent** to proceed.
   - If the connection runs on more than one Secure Agent in a group, perform one of the following steps:
     - Click **Start only on the selected Secure Agent** to proceed with the selected Secure Agent.
     - Click **Start on all the Secure Agents in the group** to proceed with the Secure Agent group.

   The status of the event source is updated to **Started** on the **Connections** page.
8. To stop an event source in a connection that has been published on a Secure Agent, perform one of the following steps:
   - If the connection runs only on the selected Secure Agent, click **Stop on the selected Secure Agent** to proceed.
   - If the connection runs on more than one Secure Agent in a group, perform one of the following steps:
     - Click **Stop only on the selected Secure Agent** to proceed with the selected Secure Agent.

- Click **Stop on all the Secure Agents in the group** to proceed with the Secure Agent group.

The status of the event source is updated to **Stopped** on the **Connections** page.

Application Integration also updates the event source status on the **Event Sources** tab of the connection as shown in the following image:



For a connection that runs on the Cloud Server, a Secure Agent, or a Secure Agent group, if the status of the event source is stopped, you can also republish the connection and restart the event source. However, when you republish the connection, all the event sources in the connection start by default. In case of a Secure Agent, all the event sources in the connection start only if the Secure Agent is up and running. The status is updated on the **Event Sources** tab of the connection and on the **Connections** page in Application Integration Console. If you unpublish the connection, all the active event sources in the connection are stopped and the connection is removed from the **Connections** page in Application Integration Console.

Note: You might need to refresh the connection to view the updated status.

For more information about starting and stopping event sources in listener-based connections, see *Connectors for Cloud Application Integration*.

CHAPTER 10

Guides

Use the **Guides** page to view a list of guides that have been run. The list includes guides that completed successfully, completed with a fault, and guides that are still running.

Guides remain on the page for the following durations:

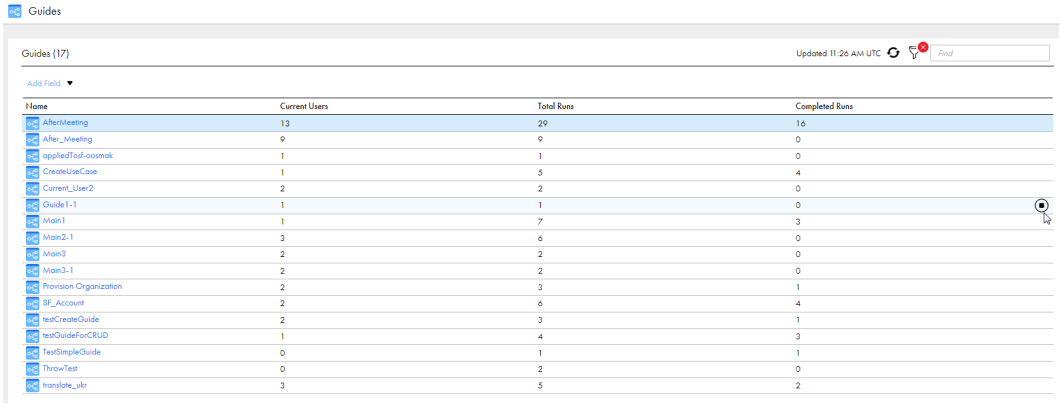
- Successful guides: 7 days.
- Faulted or prematurely terminated guides: 18 days.
- Running guides: As long as there is a running instance of the guide. After all instances are complete, it depends on whether there were successful or faulted instances.

A guide that is not running will be considered as inactive. An inactive guide will be terminated automatically after 7 days. The guide that is both inactive and that at a milestone step will be terminated automatically after 14 days.

The following information appears on the **Guides** page:

Name	Description
Name	The name that you enter in **Guide Designer** > **Start** > **General**.
Current Users	The number of users in the organization that are currently running the guide. For example, a value of 13 indicates that there are 13 instances of the guide running.
Total Runs	The number of times that users have run the guide. **Total Runs** includes successful runs, faulted runs, and on-going runs.
Completed Runs	The number of times that the guide completed successfully or with a fault.
Stop	Hover over a guide that has a non-zero **Current Users** value to see the **Stop** option. Click **Stop** to stop all instances.

The following image shows the **Guides** page:



When you click a guide name, the **Guide Results** page opens.

Guide Results Page

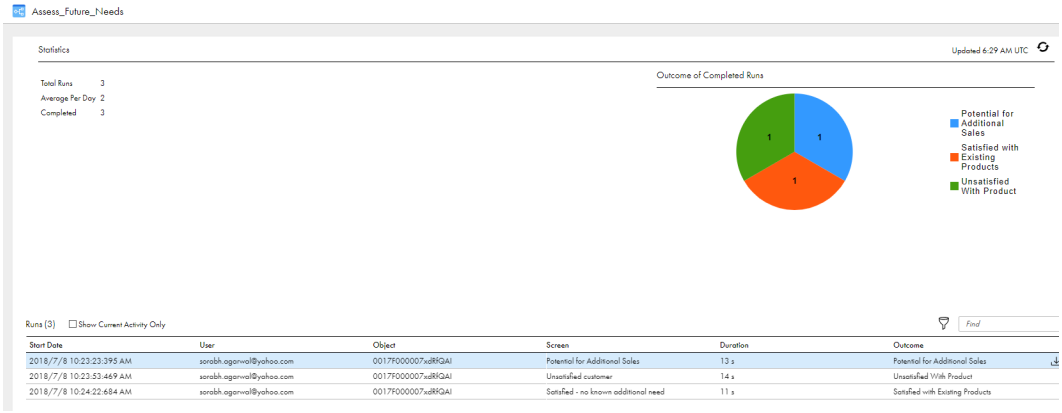
Use the **Guide Results** page to view details about guide instances.

You can view the following properties for each guide instance:

Name	Description
Start Date	The date on which the user started the guide.
User	Depends on what you defined in **Guide Designer** > **Start** > **Start** > **Applies To**. If you defined that the guide applies to **Any** object, and not a specific object, the **User** field contains the login ID of the Informatica Intelligent Cloud Services user that ran the guide. If you defined that the guide applies to a particular Salesforce object, the **User** field contains the Salesforce login ID .
Object	The ID of the Salesforce organization that contains the object that you defined in **Guide Designer** > **Start** > **Start** > **Applies To**. If you have defined that the guide applies to **Any** object, the **Object** cell does not list an object ID.
Screen	The screen at which the guide stopped. If the guide completed successfully, the name of the last screen appears. If the guide faulted, the screen at which the guide stopped appears. If the guide is running, the screen that the user is on appears.
Duration	The time taken for the guide to complete successfully or with a fault. If the guide is running, the **Duration** cell does not list a time.

Name	Description
Outcome	You see one of the following values: - An outcome that you defined in **Guide Designer** > **Start** > **Outcomes**. You see an outcome value if the guide instance completed successfully. - **Terminated Prematurely**. You see this value if the user stopped the guide before it ended. - **Unknown**. You see this value if the guide faulted. If the guide is running, the **Outcome** does not list any value.
Actions	You can download logs or stop a running guide instance.

The following image shows the **Guides Results** page:



In this image, the user ran the Asses_Future_Needs guide three times, with a different outcome each time. All instances were successful.

CHAPTER 11

Logs

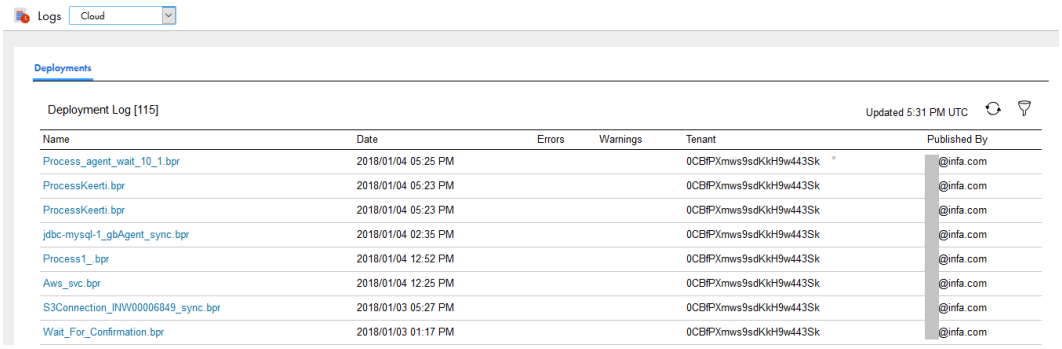
Use the Logs page to see server logs and deployment logs.

You must select a Secure Agent to see server logs.

Deployment Logs

The Deployment Logs tab shows a list of logs generated when new and modified business process archive (`.bpr`) files are deployed.

The following image shows a sample Deployment Logs tab :

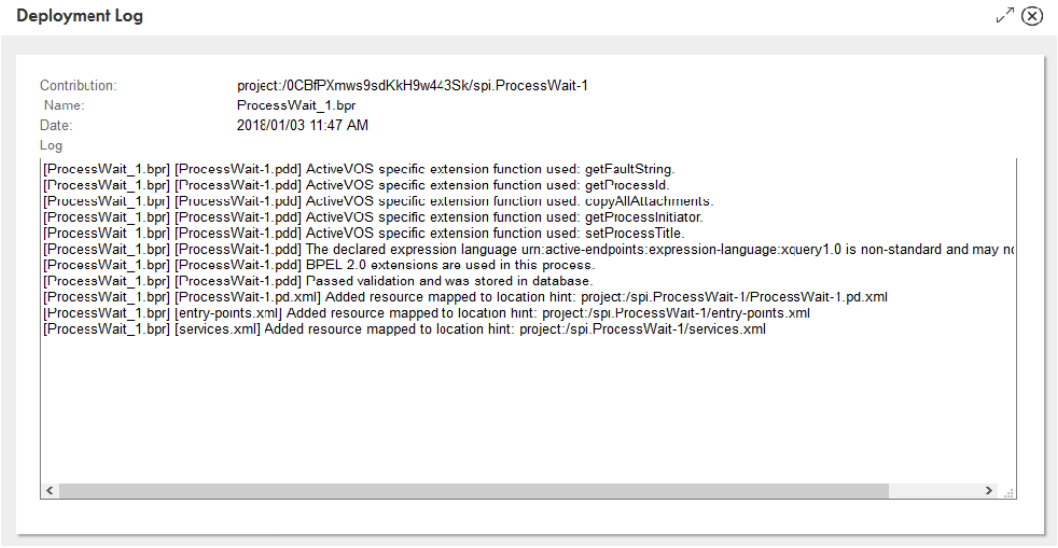


The number of errors and warnings generated, if any, are shown.

On this page you can:

- Change the display of the logs list by using the Selection Filter
- Select a `.bpr` file to view its deployment log

After you click on a `.bpr` file, Process Developer displays its Deployment Log page. This page has the name, date, and log for the selected `.bpr` file.



During deployment, the engine validates the deployment descriptor of the BPEL process, ensuring that the its WSDL file and other resources are available and valid for the current version of the process. If validation errors or warnings occur, make corrections and redeploy the `.bpr` file or create a new `.bpr` file for invalid processes.

Note: When you delete a contribution, its deployment log is also deleted.

Selection Filter

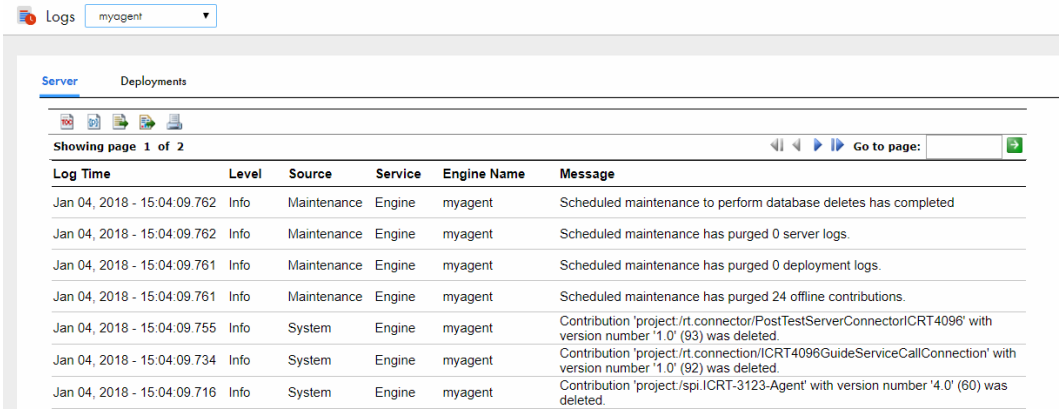
1. Select the Log Contents type, if desired.
2. Select Deployed between dates, if desired.
3. Type in the Name of a `.bpr` file, if desired. You can use a wildcard (*) in the name.
4. Select **Submit**. The Deployment Logs list redisplays based on your selection filters.

Server Log

The Server Log captures events occurring on the server so that you can view and troubleshoot problems. For each event, the log includes the time, problem level, source (initiator), service (event type), engine, and message describing the event. For example, a BPR deployment causes a log event with info level, system source, engine service.

Note: The server log may not update immediately after an event occurs. It is typically updated between 30 and 60 seconds after an event. If you are not seeing what you think should be there, you may just need to update your browser.

The following image shows a sample Server Log:



This topic discusses:

- Server Log Information
- Deleting the Process Server Log

Server Log Information

One of the following filters was used when generating logging information:

- **Log Time (Logged after and Logged before)**. By default, the log displays the most recent six hours of events. Refresh the page to view the most recent date and time. You can change the log period by setting the date and time. As a time shortcut, you can select the **Now** icon at the end of the time text box.
- **Level**. One of the following was selected:
  - Verbose—All levels
  - Info
  - Warning
  - Error
  - Critical
- **Source**. The initiator of the event or select All. Sources are:
  - System—Process Server events such as deployments
  - Maintenance—Storage and deployment log deletions
  - User—User-initiated
- **Service**. The services are:
  - Alert—Alert service configuration events
  - Email—Email service configuration events
  - Engine—Engine events, including deployments, Process Server start/stop, storage events, monitoring alerts
  - Identity—Identity service (file, LDAP, JDBC) configuration events
  - Messaging—Messaging service (JMS or other) configuration events
  - Task.—Client URI update events
  - Process—Messages added to Process Server Log from a process using the Process Server Log Comment system service
  - All—All services are shown
- **Engine Name**. The engine upon which execution occurred.

- **Message** . Text describing the event.

The console records the User/IP address when a purge or scheduling request occurs. An application server may have its own method. Here, for example, is what might be shown for Tomcat: `https://tomcat.apache.org/tomcat-7.0-doc/api/org/apache/catalina/valves/RemoteIpValve.html`.

Deleting the Server Log

Use the **Clean Up Storage** on the **Storage** tab of the **Server Configuration** page.

In addition, the interface for clearing the server log is available in Process Developer as a system service so that a developer can create and deploy a process to delete the log programmatically.

Setting the Process Server Logging Level

Use the following filters to display the Process Server log.

- **Minimum Level**. Select one of the following information levels:
  - **Verbose**. All levels
  - **Info**
  - **Warning**
  - **Error**
  - **Critical**
- **Logging Source**. Select the initiator of the event or select All. Sources are:
  - **System**. Process Server events, such as deployments
  - **Maintenance**. Storage and deployment log deletions
  - **User**. User-initiated
- **Logging Service**. Select one service or select All. The services are:
  - **Alert**. Alert service configuration events.
  - **Email**. Email service configuration events.
  - **Engine**. Engine events, including deployments, Process Server start/stop, storage events, monitoring alerts
  - **Identity**. Identity service (file, LDAP, JDBC) configuration events.
  - **Messaging**. Messaging service (JMS or other) configuration events.
  - **Task**. client URI update event.
  - **Process**. Messages added to Process Server Log from a process using the Process Server Log Comment system service.
- **Logging Engine Name**. Select one engine in a cluster or select *All*.
- **Logged after and Logged before**. By default, the log displays the most recent six hours of events. Refresh the page to view the most recent date and time. You can change the log period by setting the date and time. As a time shortcut, you can select the Now icon at the end of the time text box.
- **Message String**. Type in a text string to search for within the Message column of the log. Use wild cards to locate the message. For example, type in `Error*` to find all messages beginning with the word "Error". Type in `*-StorageException*` to find all messages containing the string "StorageException".

Deleting the Process Server Log

You can delete the Process Server log from the **Server Configuration** > **Storage** after you select a Secure Agent.

In addition, the interface for clearing the server log is available in Process Developer as a system service so that a developer can create and deploy a process to delete the log programmatically.