HCLSoftware

Unica Interact V12.1.8 Tuning Guide

Contents

Chapter 1. About tuning Unica Interact for best performance

An installation of Unica Interact consists of several components including third-party tools (such as web application servers, databases, and load balancers) and components such as Unica Platform and Unica Campaign. All of these components have several properties, features, and settings you can configure to improve performance.

Unica Interact itself has several configuration properties which you can use to tune your installation for best performance.

Defining 'best performance' is difficult. Every environment, every implementation has different requirements. For example, an implementation of Unica Interact where all data for interactive flowcharts is gathered from real-time data, would be tuned differently than an implementation requiring information read from several database tables.

Unica Interact runtime performance can be affected by many factors, including hardware configuration, network configuration, and Unica Interact configuration. The following guidelines and recommendations can have different results in your environment.

The following guidelines are organized by related components. The order in which you modify any settings does not matter.

For Simulator

- Simulator should be run in test / staging environment because simulator runs with higher number of threads and might consume a lot of CPU on Run time server.

Cache management

In a large-volume Unica Interact environment where you are using a large number of runtime servers, you can use cache management software to share the runtime load among the servers and improve the real-time performance of the runtime server group as a whole. When you enable cache management solution comes with Unica Interact you can use multiple run time servers fronted by a load balancer. The load balancer balances the workload across the runtime servers in the group, and helps to maintain something called session affinity, which means that when an incoming session is handled by runtime server A, then when the same user issues additional requests, those requests are fulfilled by the session on server A. This method of balancing connections among the runtime servers in the group can improve performance considerably, but has some limitations because all of the session information is maintained in memory, and the memory limits of each Java™ virtual machine (JVM) restrict how much session information can be maintained. Unica Interact supports below cache management solutions by default:

- **Ignite:** It is a memory-centric distributed caching solution comes with Interact. For more details about Ignite please see https://apacheignite.readme.io/docs/what-is-ignite
- **Ehcache** - It is an open source caching solution that is included with every installation of Interact. To read more about the Ehcache software, see http://www.ehcache.org/documentation/
- **Redis** - Redis is an open-source, in-memory data structure store used as a database, cache, message broker, and streaming engine. To read more about the Redis, see https://redis.io/docs/.

Working with Ehcache

To improve performance for your Unica Interact runtime server group, you can configure Ehcache by modifying a series of configuration properties in Unica Platform.

To enable Ehcache as the cache manager for each Unica Interact runtime server, set the following configuration property in Unica Platform to `EHCache`:

`Unica Interact > cacheManagement > caches > Interact cache > cacheManagerName`

To use Ehcache as the cache manager for storing event pattern states for a runtime server group, set the following parameter to `EHCache` as well:

`Unica Interact > cacheManagement > caches > PatternStateCache > cacheManagerName`

You must repeat this process on each Unica Interact runtime server in the server group to enable Ehcache as the cache manager for the server group.

After you have enabled Ehcache as the cache manager, you can configure the settings to optimize the caching for your installation.

Modifying Ehcache Configuration Settings

When you specify that an Unica Interact runtime server should use the built-in cache manager called Ehcache to improve performance, you can configure the settings used by Ehcache to optimize its value to your runtime server group.

To configure the Ehcache cache manager, you can open the following configuration properties in Unica Platform:

`Unica Interact > cacheManagement > Cache Managers > EHCache > Parameter Data`

This configuration category contains a set of default configuration properties for Ehcache that correspond to the settings you can specify in an Ehcache configuration file. You can also create additional parameters in this category by clicking ***(Parameter)*** and naming it to match the Ehcache parameter you want to modify.

For information on the configuration properties for Ehcache, see the *Unica Interact Administrator's Guide*, *Appendix B: Unica Interact runtime environment configuration properties*. You can also refer to the Ehcache documentation found at http://www.ehcache.org/documentation/.

Note that for optimal performance with Ehcache, in the Unica Platform configuration settings for the Unica Interact runtime server, set the session timeout (`Interact > cacheManagement > caches > InteractCache > TimeoutInSecs`) to the smallest acceptable value.

Each Unica Interact session contains some amount of session data in memory. The longer you maintain sessions, the more concurrent memory requirements you have. For example, if you are expecting 50 sessions per second, and each session can remain active for 20 minutes, you might require the memory to support 60,000 sessions at a time, if every session lasted the full 20 minutes.

The value must be logical for your scenario. For example, a call system session might need to remain active for a minute, but a website session should remain active for 10 minutes.

Supporting a greater number of concurrent sessions in Ehcache

When you are using Ehcache as the cache manager in some Unica Interact environments, a high number of concurrent sessions may cause the Unica Interact runtime to exceed its available memory, causing a system slowdown or out-of-memory error. An out-of-memory situation is more likely if you have increased the `maxEntriesInCache` configuration parameter (`Unica Interact > cacheManagement > Caches > InteractCache > maxEntriesInCache`) to a number higher than the

default setting, although it can occur even with the default setting of a maximum of 100,000 sessions. To avoid this issue, you can reduce the **maxEntriesInCache** value, or you can follow the instructions here to modify the system memory cache to roll the cached memory over to disk storage. This modification allows many more concurrent sessions than would otherwise be possible.

To prevent the Unica Interact runtime from exceeding the available memory in the Java™ virtual machine (JVM) memory heap, you can modify the memory caching mechanism to use disk storage for caching the data that exceeds the available memory.

System administrators can adjust the amount of memory available to the deployment systems via the following JVM parameters:

-Xms####m -Xmx####m -XX:MaxPermSize=256m

Where the characters #### should be 2048 or higher (depending on their system load.) Note that a 64-bit application server and JVM are usually necessary for values greater than 2048.

Unica Interact uses an open source distributed caching system called Ehcache for caching data. By default, Unica Interact uses the settings specified by the Unica Platform to manage the Ehcache caching. However, you can override those settings for Unica Interact by creating your own Ehcache configuration file that is automatically loaded whenever Unica Interact starts up.

To load a custom Ehcache configuration file on startup, the following must be true:

- Your JVM must include the parameter `interact.ehcache.config` property, as in the following example:

  `-Dinteract.ehcache.config=/temp/abc.xml`

  You can set a JVM property for your web application server in the startup command script (Oracle WebLogic) or Admin Console (IBM® WebSphere®). The information in */temp/abc.xml* is the actual path to the XML file containing the Ehcache configuration you want to load at startup.
- A configuration file containing valid Ehcache configuration settings in XML format must exist at the location specified by the JVM property.

  If you do not set this property, or if you set this property and there is no configuration file at the specified location, Unica Interact uses its default cache configuration.

If both conditions are true, the Ehcache configuration file is loaded on startup, and its settings override any default Unica Interact configuration parameters for caching session data.

The following example shows a sample configuration file (in XML format) that you might use to customize Ehcache:

```
<config
      xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xmlns='http://www.ehcache.org/v3'
      xmlns:eh='http://www.ehcache.org/v3'
      xmlns:jsr107='http://www.ehcache.org/v3/jsr107'>
   <service>
      <jsr107:defaults enable-statistics="true" enable-management="true"/>
   </service>

   <persistence directory="${ehcache.disk.store.dir}"/>
```

```
   <cache alias="InteractCache">
      <key-type>java.lang.String</key-type>
      <value-type>com.unicacorp.interact.session.InteractSession</value-type>
      <expiry>
         <tti unit="seconds">300</tti>
      </expiry>
      <listeners>
         <listener>
            <class>com.unicacorp.interact.cache.ehcache.EHCacheEventListener</class>
            <event-firing-mode>ASYNCHRONOUS</event-firing-mode>
            <event-ordering-mode>UNORDERED</event-ordering-mode>
            <events-to-fire-on>CREATED</events-to-fire-on>
            <events-to-fire-on>UPDATED</events-to-fire-on>
            <events-to-fire-on>EXPIRED</events-to-fire-on>
            <events-to-fire-on>REMOVED</events-to-fire-on>
            <events-to-fire-on>EVICTED</events-to-fire-on>
         </listener>
      </listeners>
      <resources>
         <heap unit="entries">10000</heap>
         <!--<offheap unit="MB">100</offheap>-->
         <disk persistent="true" unit="GB">1</disk>
      </resources>
   </cache>
   <cache alias="PatternStateCache">
      <key-type>com.unicacorp.interact.treatment.optimization.IAudienceID</key-type>
      <value-type>com.unicacorp.interact.eventhandler.eventpatterns.state.PatternStatesForAudience</value-type>
      <expiry>
         <tti unit="seconds">600</tti>
      </expiry>
      <loader-writer>
         <class>com.unicacorp.interact.cache.ehcache.loaderwriter.InteractCacheLoaderWriterAdapter</class>
         <write-behind>
            <batching batch-size="100" coalesce="true">
               <max-write-delay unit="seconds">5</max-write-delay>
            </batching>
         </write-behind>
      </loader-writer>
      <listeners>
         <listener>
            <class>com.unicacorp.interact.cache.ehcache.EHCacheEventListener</class>
            <event-firing-mode>ASYNCHRONOUS</event-firing-mode>
            <event-ordering-mode>UNORDERED</event-ordering-mode>
            <events-to-fire-on>CREATED</events-to-fire-on>
            <events-to-fire-on>UPDATED</events-to-fire-on>
            <events-to-fire-on>EXPIRED</events-to-fire-on>
            <events-to-fire-on>REMOVED</events-to-fire-on>
            <events-to-fire-on>EVICTED</events-to-fire-on>
         </listener>
      </listeners>
      <resources>
         <heap unit="entries">10000</heap>
         <!--<offheap unit="MB">100</offheap>-->
         <disk persistent="true" unit="GB">1</disk>
      </resources>
   </cache>
</config>
```

If you saved this file as `/HCL/Unica Interact/conf/Ehcache.xml`, you would then set the JVM property for the web application as in the following example:

```
-Dinteract.ehcache.config=/HCL/Unica Interact/conf/Ehcache.xml
```

For a complete list of the options available for modifying the Ehcache software, see http://www.ehcache.org/documentation/

Working with Ignite

To improve performance for your Unica Interact runtime server group, you can configure Ignite by modifying a series of configuration properties in Unica Platform.

To enable Ignite as the cache manager for each Unica Interact runtime server, set the following configuration property in Unica Platform to `Ignite`:

```
Unica Interact > cacheManagement > caches > Interact cache > cacheManagerName
```

To use Ignite as the cache manager for storing event pattern states for a runtime server group, set the following parameter to `Ignite` as well:

```
Unica Interact > cacheManagement > caches > PatternStateCache > cacheManagerName
```

You must repeat this process on each Unica Interact runtime server in the server group to enable Ignite as the cache manager for the server group.

After you have enabled Ignite as the cache manager, you can configure the settings to optimize the caching for your installation.

Modifying Ignite Configuration Settings

When you specify that an Unica Interact runtime server should use the built-in cache manager called Ignite to improve performance, you can configure the settings used by Ignite to optimize its value to your runtime server group.

To configure the Ignite cache manager, you can open the following configuration properties in Unica Platform:

```
Unica Interact > cacheManagement > Cache Managers > Ignite > Parameter Data
```

This configuration category contains a set of default configuration properties for Ignite that correspond to the settings you can specify in an Ignite configuration file. You can also create additional parameters in this category by clicking ***(Parameter)*** and naming it to match the Ignite parameter you want to modify.

For information on the configuration properties for Ignite, see the *Unica Interact Administrator's Guide*, *Appendix B: Unica Interact runtime environment configuration properties*. You can also refer to the Ignite documentation found at https://apacheignite.readme.io/docs.

Note that for optimal performance with Ignite, in the Unica Platform configuration settings for the Unica Interact runtime server, set the session timeout (`Interact > cacheManagement > caches > InteractCache > TimeoutInSecs`) to the smallest acceptable value.

Each Unica Interact session contains some amount of session data in memory. The longer you maintain sessions, the more concurrent memory requirements you have. For example, if you are expecting 50 sessions per second, and each session can

remain active for 20 minutes, you might require the memory to support 60,000 sessions at a time, if every session lasted the full 20 minutes.

The value must be logical for your scenario. For example, a call system session might need to remain active for a minute, but a website session should remain active for 10 minutes.

Supporting a greater number of concurrent sessions in Ignite

When you are using Ignite as the cache manager in some Unica Interact environments, a high number of concurrent sessions may cause the Unica Interact runtime to exceed its available memory, causing a system slowdown or out-of-memory error. An out-of-memory situation is more likely if you have increased the `maxEntriesInCache` configuration parameter (`Unica Interact > cacheManagement > Caches > InteractCache > maxEntriesInCache`) to a number higher than the default setting, although it can occur even with the default setting of a maximum of 100,000 sessions. To avoid this issue, you can reduce the **maxEntriesInCache** value, or you can follow the instructions here to modify the system memory cache to roll the cached memory over to disk storage. This modification allows many more concurrent sessions than would otherwise be possible.

To prevent the Unica Interact runtime from exceeding the available memory in the Java™ virtual machine (JVM) memory heap, you can modify the memory caching mechanism to use disk storage for caching the data that exceeds the available memory.

However, enabling disk storage will cause a performance degradation. System administrators can adjust the amount of memory available to the deployment systems via the following JVM parameters:

-Xms####m -Xmx####m -XX:MaxPermSize=256m

where the characters #### should be 2048 or higher (depending on their system load.)

Note: A 64-bit application server and JVM are usually necessary for values greater than 2048.

Failover

When distributed Ignite is used, the data is saved across the whole grid. If the instance hosting a session is stopped, that session data will be lost and all subsequent requests for that session will fail. To achieve failover, you need to set parameter numberOfBackups under Affinium|Interact|cacheManagement|Cache Managers|Ignite|Parameter Data. Its value is the number of backup copies of the data in the cache. A higher value will have better failover protection and better read performance with the cost of lower write performance. Therefore, it should be chosen based on the use cases.

Unica Interact API

Use the Java™ Serialization API instead of SOAP API. The Serialization API provides better throughput (can be 5-10 times more depending on the application configuration) and a shorter response time.

For information on implementing a custom Java API timeout, see the API documentation installed with Unica Interact at *<Interact_home>/docs/apiJavaDoc* or see the sample code and comments in *<Interact_home>/samples/api/SampleCustomizedInteractAPI.java*.

Installation and network configuration

If the Unica Interact server is required to maintain session data across multiple Unica Interact API calls and you are using the Ehcache cache manager to improve performance, use sticky load balancing and local session management instead of distributed session management.

Using Ehcache, distributed mode incurs the cost of communication between the Unica Interact runtime servers to keep the sessions consistent. Local session management can avoid that cost.

In Unica Platform configuration settings for the Unica Interact runtime server, set the `Interact > cacheManagement > EHCache > Parameter Data > cacheType` property to `local`.

Interactive flowchart management

Every interactive flowchart requires at least one thread to run. You can monitor a live system to see whether there are enough threads for all the interactive flowcharts.

Using JMX console, monitor the JMX statistics `CurrentJobsInProcessBoxQueue` and `CurrentJobsInSchedulerQueue` under `com.unicacorp.interact.flowchart`. Ideally, they should be zero even under peak load, which implies there are enough threads to handle the requests for flowchart executions.

Note: Running the JMX console does impact performance. You should not run the JMX console in a production environment except when diagnosing an issue.

You can control these queues with the number of threads used by interactive flowcharts. You set flowchart thread pool sizes in Unica for Unica Interact runtime under `Interact > flowchart`.

- Set `maxNumberOfFlowchartThreads` to be at least the maximum number of concurrent users expected on the Unica Interact client. For example, if the maximum number of concurrent users is 50 and each call to segmentation runs one flowchart, set `maxNumberOfFlowchartThreads` to 50.
- Set `maxNumberOfProcessBoxThreads` based on the average number of concurrent paths in the flowcharts and whether the flowcharts are CPU bound or I/O bound. It should be at least equal to `maxNumberOfFlowchartThreads`. For example, if the average number of concurrent paths in the flowcharts is 2 and all the process boxes are CPU bound, set `maxNumberOfProcessBoxThreads` to be `2*maxNumberOfFlowchartThreads`. If the process boxes are I/O bound (for example if they perform database lookups or writes, such as a Select or Snapshot process), then that number might need to be set to a larger value.
- Set `minNumberOfFlowchartThreads` to be the same as `maxNumberOfFlowchartThreads`. Likewise, set `minNumberOfProcessBoxThreads` to be the same as `maxNumberOfProcessBoxThreads`.

Service tuning

Unica Interact has several services that manage database reads and writes by various components of Unica Interact, for example, the built-in learning module and the contact and response history module.

Set threshold for each of the services (under `Interact > services > service name > cache > threshold`) to appropriate values based on the number of operations per second and the time for each insert to database. For example, if the system throughput requirements are 500 transactions per second and each transaction has 2 log contact calls, then the `contactHist` threshold should be set to a value based on the average time to write a batch and 1000 log contacts per second.

Web application server tuning

Within Unica Interact, you tune the web application by modifying JVM arguments and connections. The JVM arguments affect throughput and startup time. The number of connections you use is determined by the features you have enabled.

You should also see the documentation for your web application servers and operating systems for information about best practices for performance tuning.

JVM arguments

Java™ virtual machine (JVM) arguments should be defined using startup command script or Admin Console for your web application server.

- Confirm you have the latest service packs and patches installed for your operating system, web application server, and JVM.
- For best performance, when using Sun HotSpot VM, use the `-server` argument.
- Decide on a maximum heap size for the JVM, based on the memory availability in the server. (Unica Interact is not a memory intensive application). Set the max and min sizes of the heap to be the same (using `-Xmx` and `-Xms` arguments), which increases the startup time, but gives better throughput.
- If the application is unresponsive periodically, for example, long response times running to few seconds, the Garbage Collection policy might need to be tuned. Monitor Garbage Collection runs using JMX console and by studying the Garbage Collection output after enabling the following arguments.

  ```
  -verbosegc -XX:+PrintGCDetails
  ```
- In our tests, the Low Pause Collector was found to eliminate Garbage Collection-related slowness without sacrificing throughput. The following is one set of options was found to be useful for a 2 GB JVM heap.

  ```
  -XX:+UseConcMarkSweepGC -Xmn512m -XX:SurvivorRatio=6
  ```

  In general, the young collection should be about 1/4 to 1/2 of the total heap. The Survivor Space can be set to 1/8th the size of young collection.
- If you use a two-digit year (for example, 01-01-20) or you use dates that are on or after 01/01/2020 when you use a Date macro, you must add the following JVM parameter to the application start so the two-digit year is fixed to a four-digit year, which is what is expected by the application.

  ```
  -DInteract.enableTwoDigitYearFix=true
  ```
- Under certain circumstances, deploying older legacy interactive channels or interactive channels with large deployment histories can stress the system and require 2048mb or greater of Campaign designtime and/or Interact runtime Java heap space.

  System administrators can adjust the amount of memory available to the deployment systems via the following JVM parameters:

-Xms####m -Xmx####m -XX:MaxPermSize=256m

Where the characters #### should be 2048 or higher (depending on their system load.) Note that a 64-bit application server and JVM are usually necessary for values greater than 2048.

References

- Tuning Garbage Collection with the 5.0 Java™ virtual machine (http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html)
- Java™ Tuning white paper (http://java.sun.com/performance/reference/whitepapers/tuning.html)

Connection pool

Set the size of the connection pool of the Unica Interact runtime data source using the application server console. Take into account the number of concurrent users and the connections made during the lifetime of a session, which includes loading profile, loading offer suppression, reads and writes from flowcharts, and reads from learning.

Feature/Option	Connections Required If Enabled
At least one of the following features is enabled • Load profile table • Load Offer Suppression table • Load Score Override table	1 connection per concurrent client call to startSession or setAudience It does not matter if only one table load or all three table loads are enabled.
Learning	2 connections
At least one logging or tracking service enabled	The value of `Interact > services > threadManagement > flushCacheToDB > maxPoolSize`. The default is 5.
Flowcharts that make at least 1 database call	The value of `Interact > flowchart > maxNumberOfFlowchartThreads`. The default is 25.

For example, if you have the following requirements.

- Require that 30 concurrent calls to `startSession` not wait when obtaining a database connection (30)
- Have learning turned on (2)
- All services turned on (5)
- Have at least one deployed flowchart that makes a db connection (25)
- Rely on current defaults (0)

Then you should set up a database connection pool size with a minimum of 62 (30+2+5+25) for optimal performance where no single consumer of the connection will wait.

Tuning terminology

System tuning has specific definitions for common terms.

Response time

The amount of time it takes for the Unica Interact runtime server to respond to an API request as measured from the client side.

Throughput

The number of transactions per second.

Transaction

A call to the Unica Interact runtime server by the Unica Interact API, including calls that are defined by the InteractAPI class such as `startSession` and `setAudience`. The `executeBatch` call is one transaction, even though it can contain several commands. These do not include methods that work with response objects, for example, the Offer class.

Database tuning

Database tuning involves adding indexes to specific tables and updating statistics.

Add appropriate indexes in Profile, Offer suppression, and Score override tables.

- Profile tables. Create a unique index on the audience level fields.
- Offer suppression tables. Create an index on the audience level fields.
- Score override tables. Create an index on the audience level fields.

Also, make sure that the statistics on these indexes are up to date. For example, if the Audience ID is a combination of two columns `CustomerId` and `HouseholdId`, create an index on these columns in all the tables and update the statistics.

ETL Tuning

When you configure the contact and response history module, the module uses a background Extract, Transform, Load (ETL) process to move data from the runtime staging tables to the Unica Campaign contact and response history tables.

This section describes the optional configuration settings you might want to change in Unica Interact to improve the performance of the ETL tool. You might not need to modify any of these configuration parameters from their default settings; however, if you do, follow the guidelines here and in the *Unica Interact Administrator's Guide* to modify the tool's performance.

All of the properties described here are found in Unica Campaign configuration properties, in `Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking`.

Configuration Property	Value and Description
`processSleepIntervalInMinutes`	The number of minutes the Unica Interact contact and response history module waits between copying data from the Unica Interact runtime staging tables to the Unica Campaign contact and response history tables. The default value is 60.

Configuration Property	Value and Description
`purgeOrphanResponseThresholdInMinutes`	This property determines how long Unica Interact waits before purging responses that have no corresponding contact (also known as "orphaned responses"). The default is 180; however, for processing many records, the delay between processing contacts and responses may be greater, and you would therefore increase this value to prevent responses from being purged too quickly.
`maxJDBCInsertBatchSize`	Out of the total number of records that the contact and response history module processes in one iteration, this is the maximum number of records of a JDBC batch to process (and assemble together in a batch) before committing the query into the Unica Campaign system tables The default value is 1000. Because this value works together with the `maxJDBCFetchBatchSize` property, you might need to increase this value if that property also increased significantly. For example, if you set `maxJDBCFetchBatchSize` to 2,500,000, you might increase this value to 10,000 to handle the increase in records. Note that memory requirements increase as you increase this value; a setting of 10,000 for this property is a good upper-limit due to memory demands.
`maxJDBCFetchBatchSize`	Determines the maximum number of records to fetch from the staging database for an ETL batch processing operation. The default value is 1000, but to tune the performance of the contact and response history module, make sure to set this value to a number greater than the number of contact history records generally processed each day. This property is used together with `maxJDBCFetchChunkSize` and `maxJDBCInsertBatchSize` to determine how the records are processed. For example, suppose the values were set as shown here: • `maxJDBCFetchBatchSize`: 30000 • `maxJDBCFetchChunkSize`: 1000 • `maxJDBCInsertBatchSize`: 1000

Configuration Property	Value and Description
	In this example, 30,000 records are fetched (or the total number of records if there are under 30,000). Then, the contact and response history module loops through that 30,000 records, processing 1,000 at a time, so that 1,000 records are marked in the staging tables, and 1,000 are inserted into the detail contact history table.
`maxJDBCFetchChunkSize`	Determines the maximum number of records in a JDBC chunk (from a total of up to `maxJDBCFetchBatchSize` records) to process with each pass The default value is 1000. In some cases, you might be able to improve performance by increasing this value above the `MaxJDBCInsertBatchSize` property value.
`deleteProcessedRecords`	This property, which specifies whether to retain contact and response history records after they are processed, is set to `YES` by default. Changing this value can give you more control over the data flow within the ETL process and affect performance (by delaying the purging of these records until a later time of your determination); however, you must be knowledgeable to handle the maintenance of these records manually to be sure that they are removed at appropriate times. Use caution when modifying this setting.
`fetchSize`	Providing a value for the JDBC `fetchSize` can improve performance for large batches of records, but the tradeoff for improved networking performance is the impact of larger fetch sizes on memory usage. See the description of this configuration property in the *Unica Interact Administrator's Guide* for more information about adjusting this setting.

For detailed descriptions of each of the configuration properties described here, see the online help for that configuration page, or see the *Unica Interact Administrator's Guide.*

Logging

Make sure the log level is set to INFO or ERROR. Never use a verbose log setting like DEBUG or TRACE in a production environment.

There are three places where you can configure logging:

- Set the logging level in the `interact_log4j.properties` file. By default, this file is installed in `<install_dir>/Interact/conf` directory, where <*install_dir*> is the parent directory where your products are installed.
- Confirm that Unica Interact API is not logging. Logging is determined by Boolean setting available in the `startSession` and `setDebug` methods.
- Confirm that JMX monitoring is set to Info with the `activateInfo` JMX operation.