INTRODUCTION TO THIS GUIDE

TECHNICAL CONFIGURATION

1 Introduction

1.1 Summary of Configuration Options

2 JBoss Configuration Procedures

2.1 Standard Procedure for Making Configuration Changes

3 General Configuration Options

3.1 JBoss JVM Parameters

3.1.1 JBoss JVM Memory Size

3.1.2 JBoss Cache Configuration (5.4 SP1-CU05+)

3.1.3 Enabling Garbage Collection Logging for the JBoss JVM

3.2 Claim Processing Threads

3.3 Java EE Transaction Architecture (JTA) Timeout

3.4 Rules Install Timeout

3.5 Database Connections

3.5.1 Oracle Maximum Connection Configuration

3.5.2 Adding a Connection to a Reporting Database

3.6 Changing Database Passwords

3.6.1 Database User/Password Storage Locations

3.6.2 Password Encryption

3.7 Changing the Keystore/Truststore Passwords for HTTPS

3.8 JBoss Logging Verbosity

3.9 Claim History Retrieval

3.10 Cross Claim Type Editing

3.11 Claim Line Expansion

3.12 Force New/Updated DDR to Test/Live on KB Load

3.13 Claim Connector Configuration

3.13.1 Configuration for Multi-Session and Single-Session Clients

3.13.2 Prevention of Parallel Patient Claim Processing (pre-SP1 CU04)

3.13.3 Patient Claim Serialization (post-SP1 CU04)

3.13.4 Startup Timing

3.13.5 Handling Large Claim Messages

3.14 User Authentication Using an LDAP Server

3.15 Reduce Claim Purge DB Blocking

4 Multi-Node Claim Processing with High Availability

4.1 Configuration for JMS Server Nodes

4.1.2 Configuring the broker.xml File
4.1.3 Configuring the jndi.properties File ................................................................. 48
4.2 Configuration that Applies to All Nodes ........................................................... 49
  4.2.1 Configuring the standalone.xml File ............................................................. 49
4.3 Configuration for “JBoss Only” Nodes ............................................................... 51
  4.3.1 Configuring the services.properties File ....................................................... 52
4.4 Non-HA Configuration ..................................................................................... 52
  4.4.1 Configuring the standalone.xml File ............................................................. 52
  4.4.2 Configuring the services.properties File ....................................................... 53
4.5 Considerations for IPv6 ................................................................................... 53
  4.5.1 Disabling IPv6 .............................................................................................. 54
4.6 Web Session Load Balancing .......................................................................... 54
  4.6.1 STATIC Load Balancing Methods ................................................................. 54
  4.6.2 DYNAMIC Load Balancing Methods ............................................................ 54
  4.6.3 Sticky Session/Session Affinity ................................................................... 55
4.7 Operational Considerations ........................................................................... 55
  4.7.1 Location of JMS Journal Files ....................................................................... 55
  4.7.2 Operating an HA System ............................................................................. 55

5 Setting up Secure Communications .................................................................... 57
  5.1 Introduction to Secure Communication ........................................................... 57
    5.1.1 Secure Communication Protocols ............................................................... 57
    5.1.2 Digital Certificates and Keys ....................................................................... 57
    5.1.3 Keystore and Truststore ............................................................................. 58
    5.1.4 The Java Keytool Command ....................................................................... 59
    5.1.5 Keytool Summary ..................................................................................... 59
  5.2 Getting Started ................................................................................................ 59
    5.2.1 Upgrade Considerations ............................................................................ 60
    5.2.2 Distributing Signed Certificates ................................................................. 61
    5.2.3 Using a Workspace Directory ...................................................................... 62
    5.2.4 Summary of Parameters for the Keytool Utility ....................................... 62
  5.3 Managing Digital Certificates ......................................................................... 64
    5.3.1 Obtain a Private/Public Key Pair ................................................................. 64
    5.3.2 Import the Signed Certificate into the Keystore ....................................... 66
    5.3.3 Distributing Signed Certificates ................................................................. 68
  5.4 Updating the Keystore for Cluster Nodes ...................................................... 69
    5.4.1 Deploying the Keystore File ...................................................................... 70
  5.5 Securing the JBoss Application Server (All nodes) ........................................ 70
    5.5.1 Configuring the Standalone.xml File ........................................................ 70
    5.5.2 Securing Remote EJB Access (jboss-ejb-client.properties) ..................... 74
    5.5.3 Configuring the ICP Service for HTTPS ................................................... 75
    5.5.4 Securing JMS Communication ................................................................. 76
    5.5.5 Configuring HTTPS with High Availability Enabled ............................... 80
    5.5.6 Configuring Rules Install for Secure HTTP (DB update) ......................... 80
    5.5.7 Restarting Services ................................................................................... 81
© 2020 Optum, Inc.
6 Enabling ACE Reporting User Interface 82

COMMAND-LINE TOOLS

1 Load KnowledgeBase 83
2 Bulk Load LCD Carriers 84
3 Password Encryption 85
4 Configuration Saver 86
5 Configuration Loader 87

SUPPORT TOOLS

1 DDR Trace Instructions 88
  1.1 Requirements ............................................................................................................. 88
  1.2 To Run a Trace within DDR ....................................................................................... 88
  1.3 Reading Available Trace Data .................................................................................. 89

2 Configuration Capture and Loader 90
  2.1 Overview ..................................................................................................................... 90
  2.2 Requirements and Methods of Running the Tool ................................................... 90
  2.3 URL/Support Page Method ....................................................................................... 91
  2.4 Configuration Saver and Loader Tools ................................................................... 95
    2.4.1 Recommendations and Requirements ................................................................. 96
    2.4.2 Capturing a Configuration .................................................................................... 96
    2.4.3 Loading a Saved Configuration ............................................................................ 99
  2.5 Configuration Capture/Load Via the Command-Line ........................................... 102
    2.5.1 Capturing a Configuration (command-line) ...................................................... 102
    2.5.2 Loading a Configuration (command-line) ......................................................... 103
  2.6 Common Use Cases ................................................................................................ 104
    2.6.1 Creating a test environment from a production environment ......................... 104
    2.6.2 Updating a production environment from a test environment ....................... 105
    2.6.3 Best Practices for updating production environments ................................ 105
  2.7 Claim Migration Tool ............................................................................................. 105
    2.7.1 Recommendations and Requirements ............................................................. 106
    2.7.2 Running the Claim Migration Tool ........................................................................ 106

3 Performance Monitor 109
  3.1 Introduction ............................................................................................................. 109
  3.2 Prerequisites ........................................................................................................... 109
3.3 Installation and Quick Start ................................................................. 109
3.4 SSL Setup ........................................................................................... 111
3.5 CPM/LPM Metrics ............................................................................... 113
3.6 Connectors Table ................................................................................. 113
3.7 Recent Interesting Claims ................................................................. 114
3.8 View Claim Details .............................................................................. 115
  3.8.1 Performance View ........................................................................... 116
  3.8.2 Cluster Performance Graph .............................................................. 116
  3.8.3 Node Comparison Graph ................................................................. 117
  3.8.4 Average Throughput Time ............................................................... 118
  3.8.5 Advanced View ............................................................................... 120

**SQL TOOLS**

1 Report Queries ...................................................................................... 122
  1.1 How to Run the Report Queries .......................................................... 122
  1.2 Rule in Ruleset Reports ...................................................................... 122
    1.2.1 ILOG Rules in Ruleset Query (Includes flag status) ...................... 122
    1.2.2 DDR Rules in Static Ruleset Query (Includes flag status) .............. 123

2 Delete Inactive LCDs in Bulk ................................................................. 125

3 Remove Calendar Month LCD Frequencies ....................................... 126

4 Database User Permissions .................................................................. 128
  4.1 Product Admin and Product User Permissions (Oracle) .................... 129
    4.1.1 Managing Privileges for the Product Admin User (Oracle) ........... 131
  4.2 Product Admin and Product User Permissions (SQL Server) .......... 132

Glossary ....................................................................................................... 134

Document Revision History ..................................................................... 138
Introduction to This Guide

This document includes information about the following functionality in the CM/CES product suite:

- **Technical Configuration**
- **Configuration Capture and Loader**
- **Performance Monitor**
- **SQL Tools**

If you have questions about configuring or using these tools, contact Optum Technical Support at 1-800-765-6818, option 1.
1 Introduction

This section describes the configuration options most likely to be beneficial in tuning the performance of the CM/CES software and adjusting the logging system to gain more or less visibility into internal operations.

The internal application architecture allows for true High Availability when multiple claim processing nodes are configured. Previously, JBoss Application Server clustering was employed, but this has been replaced in favor of only clustering the Java Messaging Service message brokers. See Multi-Node Claim Processing with High Availability for information about configuring High Availability.
1.1 Summary of Configuration Options

This table summarizes the configuration options that must be tailored for each installation of CM/CES software to make the best use of the computer hardware. Corresponding sections below provide the details of how to modify them. There are many other configuration settings documented below that may need to be changed in some circumstances, but it is expected that for most installations, the “out of the box” default values will work well, so they are not included in this table.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>File</th>
<th>Restart</th>
<th>Default</th>
<th>Suggested</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss JVM memory size (non-clustered)</td>
<td>jboss.wrapper.conf</td>
<td>JBoss</td>
<td>4 GB</td>
<td>4 to 6 GB. <strong>NOTE:</strong> If the DDR LCD rule is used, then a minimum of 6GB is required!</td>
</tr>
<tr>
<td>Claim processing threads (MDBs, rule engines)</td>
<td>standalone.xml</td>
<td>JBoss</td>
<td>15</td>
<td>CPU count X 2 with a minimum of 15.</td>
</tr>
</tbody>
</table>

**Note** The Restart column indicates the service that must be restarted for the configuration option to take effect. Also note that in the case of the JBoss service, it is probably easiest to stop and restart all CM/CES services using desktop icons rather than restarting individual services.
2 JBoss Configuration Procedures

All configuration settings for JBoss are contained in the standalone.xml file. Two copies of this file are kept in two different locations:

| Master Folder | <InstallDir>\jboss\icpConfig\standalone\ |
| Runtime Folder | <InstallDir>\jboss\standalone\configuration\ |

This is because JBoss sometimes reformats the file when it starts, stripping all comments. If the file has an XML syntax error, it will also be reformatted and comments stripped.

2.1 Standard Procedure for Making Configuration Changes

The following steps should be followed when making changes to the JBoss configuration file (standalone.xml):

In the “master” folder (<InstallDir>\jboss\icpConfig\standalone\), make a backup copy of standalone.xml. It is recommended to use a name that identifies what it was prior to the intended changes. (Example: standalone.xml.preClustering_20170321).

Modify standalone.xml with the desired configuration changes.

Stop CM/CES services.

Copy standalone.xml from the “master” folder to the “runtime” folder (<InstallDir>\jboss\standalone\configuration\), overwriting the existing file.

Start CM/CES services.

Check the logs to verify that everything started correctly.
3 General Configuration Options

The following subsections provide details on modifying configuration settings that apply generally to all CM/CES implementations. Configurations specific to installations that employ JMS Server clustering are covered in Multi-Node Claim Processing with High Availability.

3.1 JBoss JVM Parameters

3.1.1 JBoss JVM Memory Size

**Background:** The maximum amount of memory for program data storage (not the Java code) is a startup command line parameter to the Java Virtual Machine (JVM) that runs the JBoss JEE application server. Because the CM/CES product runs JBoss as a Windows service, the JVM parameters are specified in a service wrapper configuration file.

**Recommendations:** Although the software will run with as little as 4 GB allocated to the JVM, this can result in frequent memory garbage collection which can slow performance. Optum recommends that up to 8 GB be assigned if enough memory is available on the system.

**Note**
If the DDR version of the LCD rule is used, a large amount of LCD data is cached to improve performance. In this case, a minimum of 6 GB must be allocated. It is also recommended that the server on which the application is running have at least 16 GB of system memory (RAM).

**Configuration Example:** Search for `-Xmx` and modify the associated value.

**Configuration File Location**

`<InstallDir>/conf/jboss.wrapper.conf`

```
# Java Additional Parameters
wrapper.java.additional.1=-Xms2g
wrapper.java.additional.2=-Xmx6g
wrapper.java.additional.3=-Xss2m
wrapper.java.additional.4=-XX:+UseG1GC
wrapper.java.additional.5=-XX:-TieredCompilation
wrapper.java.additional.6=-Djava.net.preferIPv4Stack=true
wrapper.java.additional.7=-Dhibernate.config=apollo.oracle.cfg.xml
```

3.1.2 JBoss Cache Configuration (5.4 SP1-CU05+)

**Background:** In order to achieve high performance rule execution, the most frequently used KnowledgeBase data is loaded into cache memory as claims are processed. Many of the caches are initially empty, and KnowledgeBase values are added the first time they are read from the database and are available in cache memory from that point on.

Memory requirements are high due to the size of the KnowledgeBase and the need to replicate the cache across enterprises/rulesets. To ensure that the cache does not grow and consume all memory, the caching mechanism was changed as of 5.4 SP1-CU05 so that cache entries that have been idle for more than a specified number of hours are automatically removed from the cache. By default, the timeout is 12 hours.
Recommendations: Unless there is evidence that a JVM is low on memory (high CPU utilization and low throughput), the default configuration can be used. However, when it has been determined that too much memory is being consumed, the timeout value for clearing idle cache entries should be reduced to 1 or 2 hours. See section 0 below for further details.

Configuration Example: Search for `cacheEntry` and modify the associated value.

### Configuration File Location

```
<InstallDir>/conf/jboss.wrapper.conf
```

```java
# Java Additional Parameters
: wrapper.java.additional.17=-Djavax.net.ssl.trustStorePassword=ingenix1
wrapper.java.additional.18=-Dorg.tanukisoftware.wrapper.WrapperSimpleApp.waitForStartMain=TRUE
wrapper.java.additional.19=-Dorg.tanukisoftware.wrapper.WrapperSimpleApp.maxStartMainWait=60
#wrapper.java.additional.20=-Djdk.tls.ephemeralDHKeySize=2048
wrapper.java.additional.21=-DcacheEntryTimeToLiveInHours=12
```

3.1.3 Enabling Garbage Collection Logging for the JBoss JVM

**Background:** High CPU utilization and low throughput may indicate the JVM is low on memory. To confirm, add Garbage Collection (GC) logging to the JVM startup parameters. The log output will identify the frequency of GC events and the amount of free memory after the garbage collection is performed.

**Recommendations:** Use the GC logs as a debugging tool. Disable GC logging when debugging is complete.

**Configuration Example:** Search for `timezone` in the `jboss.wrapper.conf` file.

1. Add the lines for **GARBAGE COLLECTION LOGGING** as shown below.
2. Change the `sequence number` values as necessary to ensure each parameter is listed in sequence. Ignore any parameters that were previously commented out (‘22’ in the example below).
3. Change the file `path` to the location where log files should be written. Optum recommends writing logs to the same directory with the rest of the CM/CES log files.
4. Optionally change the file `name`.
5. Optionally change parameter values to set:
   - The maximum number of GC log files to be kept.
   - The maximum file size of each GC log file.

### Configuration File Location

```
<InstallDir>/conf/jboss.wrapper.conf
```

```java
# Java Additional Parameters
: wrapper.java.additional.20=-Djdk.tls.ephemeralDHKeySize=2048
wrapper.java.additional.21=-DcacheEntryTimeToLiveInHours=12
#wrapper.java.additional.22=-Duser.timezone=US/Mountain
#----------- GARBAGE COLLECTION LOGGING -----------
wrapper.java.additional.22=-XX:+PrintGCDetails
wrapper.java.additional.23=-XX:+PrintGCDateStamps
wrapper.java.additional.24=-verbose:gc
wrapper.java.additional.25=-Xloggc:gc_stdout.log
wrapper.java.additional.26=-Xloggc:C:/Optum/ICP/log/jbossGC.log
```
Using the Garbage Collection (GC) log data: The JVM appends 0.current to the end of the GC log file name.

A sample entry from the GC log file is shown below:

```
2018-06-15T17:01:55.586-0600: 977.307: [GC pause (G1 Evacuation Pause) (young), 0.0396109 secs]
[Parallel Time: 30.2 ms, GC Workers: 4]
[GC Worker Start (ms): Min: 977307.1, Avg: 977307.2, Max: 977307.2, Diff: 0.1]
[Ext Root Scanning (ms): Min: 1.9, Avg: 2.9, Max: 3.9, Diff: 2.0, Sum: 11.8]
[Update RS (ms): Min: 9.6, Avg: 10.3, Max: 10.8, Diff: 1.3, Sum: 41.1]
[Processed Buffers: Min: 18, Avg: 25.3, Max: 30, Diff: 12, Sum: 101]
[Scan RS (ms): Min: 0.0, Avg: 0.2, Max: 0.3, Diff: 0.2, Sum: 0.7]
[Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.2]
[Object Copy (ms): Min: 16.2, Avg: 16.6, Max: 17.0, Diff: 0.8, Sum: 66.5]
[GC Worker Other (ms): Min: 0.0, Avg: 0.1, Max: 0.1, Diff: 0.0, Sum: 0.2]
[GC Worker Total (ms): Min: 30.1, Avg: 30.1, Max: 30.2, Diff: 0.1, Sum: 120.5]
[GC Worker End (ms): Min: 977337.3, Avg: 977337.3, Max: 977337.3, Diff: 0.0]
[Code Root Fixup: 0.1 ms]
[Code Root Purge: 0.0 ms]
[Clear CT: 0.5 ms]
[Other: 8.8 ms]
[Choose Cset: 0.0 ms]
[Ref Proc: 7.6 ms]
[Ref Enq: 0.1 ms]
[Redirty Cards: 0.2 ms]
[Humongous Reclaim: 0.0 ms]
[Free Cset: 0.6 ms]
[Eden: 740.0M(740.0M)->0.0B(741.0M) Survivors: 18.0M->17.0M Heap: 1402.6M(1584.0M)->662.9M(1584.0M)]
[Times: user=0.14 sys=0.00, real=0.04 secs]
```

Timestamp: 2018-06-15T17:01:55.586-0600 - The time that the GC event occurred.

Elapsed Time: 977.307 - The number of seconds that have elapsed since the JVM was started. This identifies the time interval between GC events.

Time Required for GC: [Parallel Time: 30.2 ms, GC Workers: 4] – The time required to perform the garbage collection. In this example garbage collection required 30.2 milliseconds and used four worker threads in parallel. Higher time values indicate that GC activity is utilizing more CPU time and potentially impacting application performance.

Heap: 1402.6M(1584.0M) -> 662.9M(1584.0M) – Indicates heap utilization and heap size capacity before and after the GC event. The example above indicates heap size capacity was 1584mb. Memory utilization was 1402.6m before the GC event, and dropped to 662.9m after the GC event. The difference between the numbers highlighted in RED indicates the amount memory freed by the GC event. Smaller differences indicates that the system is low on memory.

3.2 Claim Processing Threads

Background: In CM/CES products, multiple claims can be analyzed in parallel. In the Legacy CM//CES product, this used multiple “Rules Engine” processes with a configurable maximum number. However, in the current Java EE version of the product, multiple threads of execution
are used. The Artemis JMS server (message broker) provides the threads which are handed off to JBoss via Message-Driven Beans (MDBs) that run in the application server. The size of the thread pool in the JMS server and the maximum number of MDBs allowed by JBoss must both be configured appropriately to adjust the number of claims that can be concurrently analyzed.

The engine.count parameter in the JBoss configuration file (standalone.xml) configures both the MDB count and the number of claim processing threads that are provided by the Artemis JMS service.

Recommendations: When CM/CES software is installed, the installer automatically detects the number of CPUs on the system and sets the engine.count value to match the number of CPUs or four (4), whichever is greater. However, the database capacity should also be factored into this calculation, which is more of an art than a science. If there are a large number of CPUs, it may overwhelm the database. If there are only a few CPUs, throughput may improve by increasing the engine.count value to be greater than the number of CPUs. A little experimentation is required to find where maximum claim processing throughput is achieved.

Note

When the product is installed, the default engine count is set to the CPU count or a minimum of 4. This usually results in a poor choice that will not achieve optimal performance; therefore, the engine.count setting should always be manually adjusted by the implementation engineer to a more appropriate value.

The minimum value should be 15 and the maximum value can only be determined through a trial-and-error process, but it should never be less than the CPU count. In other words, a system with 16 CPUs and an engine.count of 32 may run almost as fast as a system with 32 CPU and an engine.count of 64 because the database can only support queries for 32 concurrent processing threads. The system with 32 CPU may actually have better performance if the engine.count setting is reduced to around 40 (but this must be determined experimentally).

Configuration Example: After stopping CM/CES services, search for engine.count in the standalone.xml file and modify the associated value on the line below.

Configuration File Location

<InstallDir>\jboss\icpConfig\standalone\standalone.xml
3.3 Java EE Transaction Architecture (JTA) Timeout

**Background:** Normally, claim analysis should take only a few seconds. However, where there is extensive patient history, large numbers of claim lines (including lines on historical claims), and conditions that cause ILOG rules to be long-running (dropped lines, applied edits), a small number of claims may require a long time to complete. Transaction timeouts can also cause problems with the *Artemis* JMS provider. When one claim times out, many response messages for claims that are successfully analyzed may also fail to be delivered back to the *Claim Connector*.

**Recommendations:** In most cases it is recommended that the JTA transaction timeout be left at the default value of five minutes (300 seconds). In unusual cases you may try doubling or tripling the timeout to allow claim analysis to complete when debugging a specific problem.

**Configuration Example:** Search for `default-timeout` and change the value to an appropriate value in seconds (i.e., 300 = 5 minutes).

**Configuration File Location**

```xml
<InstallDir>\jboss\icpConfig\standalone\standalone.xml
```

```xml
<subsystem xmlns="urn:jboss:domain:transactions:3.0">
  <core-environment node-identifier="${jboss.node.name}"/>
    </core-environment>
  <recovery-environment socket-binding="txn-recovery-environment" status-socket-binding="txn-status-manager"/>
  <coordinator-environment default-timeout="300"/>
</subsystem>
```

3.4 Rules Install Timeout

**Background:** The ILOG rules install is a long-running process that normally takes more time than the JTA transaction timeout. Because of this, the rules install component manages its own transaction with a default timeout value of 60 minutes. This default timeout can be changed via an entry in the *CONFIG_PROPERTY* database table.

The rules install occurs in three phases, each in a separate transaction. Note that this implies the total time required for rules install can be longer than the rules install timeout since it is the sum of the time required for the three transactions.

1. Process all rule projects in the rules bundle (such as `workspace.zip`). This extracts each rules project into a working directory. This normally takes only a minute or two.

2. Import each rule project (such as `freya-ices-professional-rules.zip`) into Rules Team Server (RTS). This operation may take from 10 to 20 minutes.

3. Determine which rulesets include system rules that have been changed since the last rules install and deploy the new version of the rule(s) to those rulesets. This operation requires the most time if there are many rulesets that must be updated. This operation takes approximately one minute per ruleset.
Recommendations: The default value of 60 minutes is adequate for most users; however, it should be increased if more than 60 rulesets have been configured. A rough rule of thumb is to allow one minute per ruleset.

Configuration Example: The following sets the rules install timeout to 120 minutes.

Configuration File Location
Database: CONFIG_PROPERTY table

```sql
-- Oracle or SQL Server
update CONFIG_PROPERTY set PROPERTY_VALUE=120
where PATH='/rulesInstall' and PROPERTY_NAME='Timeout';
```

3.5 Database Connections
The maximum number of connections that JBoss will keep in its connection pool for each data source can be configured. It is important that the database also be configured to allow at least as many connections as the total size of all connection pools that are configured for JBoss.

3.5.1 Oracle Maximum Connection Configuration

Background: When the number of connections in the data source connection pools is increased, it does not help unless the database manager also supports an equal number of connections. The default SQL Server configuration is for unlimited connections, but Oracle must be appropriately configured.

Recommendations: Use the total number of data source connections (300+300+300 = 900) plus 50 to allow for additional miscellaneous connections such as schema browsers, management consoles, etc.

Configuration Example: This parameter is best set by using an SQL query editor such as SQLplus.

Configuration File Location
Oracle DB Server Parameter – SPFILE

1. Connect to the database as the SYSTEM user.
   sqlplus system/<password>@localhost/ICP
2. Check the current setting for the `processes` system parameter.
   show parameter processes;
3. Back up the `spfile` before making changes.
   create pfile from spfile;
4. Modify the value of the processes parameter.
   `Alter system set processes = 950 scope = spfile;`
5. For the change to take effect, you must stop Oracle services and restart them.
### 3.5.2 Adding a Connection to a Reporting Database

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th>This feature requires that the 2019 Q4A KnowledgeBase be installed. Database replication must include the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. All claim data-related tables and any other table that is used by reports must be replicated.</td>
</tr>
<tr>
<td></td>
<td>2. Tables used for configuration of reports and report scheduling must be replicated: REPORT, REPORT_COLUMN, REPORT_FILTER, SCHEDULED_REPORT, ARCHIVED_REPORT.</td>
</tr>
<tr>
<td></td>
<td>3. Data replication should be based on the “source” database tables, not the “target” tables. The reason for this is that applying a software update via either a Cumulative Update or a KnowledgeBase load can change the structure of tables. This structure change must be replicated to the Reporting database.</td>
</tr>
<tr>
<td></td>
<td>4. All stored procedures for reporting must be replicated. (It is probably best to just replicate all stored procedures.)</td>
</tr>
<tr>
<td></td>
<td>5. The database job scheduler must be running on the Reporting database since it will be used to manage the execution of scheduled reports.</td>
</tr>
</tbody>
</table>

**Background:** Reports can be generated from a “Reporting” database that is separate from the production database. This function assumes that the reporting database is being replicated from the production database using a tool such as Oracle GoldenGate or SQL Server Replication. The replication process may also enhance the data structure to allow it to be partitioned for improved performance. This section describes how to configure CM/CES to use the replicated database for reporting after it has been put in place.

**Recommendations:** A separate database for reporting is recommended for clients with high claim volume (large claim store) and extensive reporting requirements. The primary purpose is to offload the reporting functions to a separate database so as to prevent report generation from impacting the performance of the claim editing function on the production system.

**Configuration Example:** Because CM/CES is a JEE application, database connections are configured as “datasources” in the JBoss/Wildfly configuration file (standalone.xml). Therefore, the “Reporting” database must be added as a new datasource element within this configuration file. The new datasource must be named apollo-report-datasource in order to be recognized and used by the reporting module. Follow the steps below to add the new datasource. For clustered systems, the new datasource must be added to all nodes where JBoss is running.

**Configuration File Location**

`<InstallDir>\jboss\icpConfig\standalone\standalone.xml`

1. Using a text editor, open the `standalone.xml` at the location shown above.
2. Search for :datasources: to locate the datasource subsystem.
3. There are three `<xa-datasource>` elements (apollo-datasource, bres-datasource, and ilog-datasource). Copy the first `<xa-datasource>` element (the one with pool-name="apollo-datasource") and paste it after the last `<xa-datasource>` element. (Note that the sequence in which datasources are listed in this file does not actually matter.)
4. Make the changes that are highlighted in yellow below to configure it for the reporting datasource. The example below is for an Oracle datasource.

```xml
<xa-datasource jndi-name="java:jboss/apollo-report-datasource" pool-name="apollo-report-datasource"
    enabled="true" use-java-context="true">
    <!-- no property for xa datasource ServerName -->
    <!-- no property for xa datasource Instance -->
    <!-- no property for xa datasource DatabaseName -->
    <!-- no property for xa datasource PortNumber -->
    <xa-datasource-property name="URL">jdbc:oracle:thin:@hostname_or_IP:1521:icp</xa-datasource-property>
    <!-- no extra parameters for unicode support -->
    <!-- no extra parameters for xa tds -->
    <!-- no extra parameters for xa prepare sql -->
    <!-- no extra parameters for xa emulation -->
</xa-datasource>
```

Just insert "report-".

Replace this with the appropriate JDBC connection string for the reporting database.

A new security domain with this name must be configured in order for the connection pool to be properly managed. This is documented below.

5. A new `<security-domain>` element must be created in the security subsystem for the reporting datasource. The name of this security domain must match the name used in the `apollo-report-datasource` configuration (i.e. "encrypted-ds2" in the example above).

a. Search for :security: to locate the security subsystem, then search for `encrypted-ds` to locate the security domain to copy.

b. Copy the `<security-domain>` element (9 lines) and paste it below the lines just copied, then make the following changes.

```xml
<security-domain name="encrypted-ds2" cache-type="default">
    <authentication>
        <login-module code="org.picketbox.datasource.security.SecureIdentityLoginModule"
            flag="required">
            <module-option name="username" value="db-account-name"/>
            <module-option name="password" value="encrypted-pwd"/>
            <module-option name="managedConnectionFactoryName"
                value="jboss.jca:service=XATxCM,name=apollo-report-datasource"/>
        </login-module>
    </authentication>
</security-domain>
```

A new security domain with this name must be configured in order for the connection pool to be properly managed. This is documented below.
c. Change the name of the security domain to “encrypted-ds2”. (You can use a different name as long as it matches what is configured for the reporting datasource.)

d. Do the following if connections will need different credentials than those used to connect to the production datasource.
   i. Replace “db-account-name” with the username to be used by CM/CES when connecting to the reporting database.
   ii. Replace “encrypted-pwd” with the encrypted password string. This is obtained by using the `<InstallDir>\bin\encjboss` utility. (Further documentation for this procedure can be found in section 3.6.2 Password Encryption.)  
      Example: “Optum2019!” encrypts to “1ee45a7592e9b749973c38c56d082a5e”

e. For the `<module-option>` element with name=”managedConnectionFactoryName”, change the value to include the name of the reporting datasource as shown above.

6. Save the updated file and exit the text editor.

7. Follow the instructions in section 2: JBoss Configuration Procedures to update the runtime configuration for the server. These steps include restarting ICP services.

3.6 Changing Database Passwords

Background: For security reasons, it is required to change database passwords on a regular basis.

Claims Manager and Claims Edit System use three different database users to access the database, each with their own passwords.

<table>
<thead>
<tr>
<th>User / Schema</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM or SA</td>
<td>The database system account. (The default accounts for Oracle and SQL Server are “SYSTEM” and “sa”, respectively.)</td>
</tr>
<tr>
<td>ICP_IS</td>
<td>The “Optum System” user. The tables in this schema are used to track the current versions of the various database components of the CM/CES software (freya, freya_rules, product, metadata, coderepository, and import). These tables are updated each time a new release of the product is installed.</td>
</tr>
<tr>
<td>ICP_P</td>
<td>The “Product” user. The tables in this schema store the KnowledgeBase, Rule, and Claim data along with other data needed by the application (screen metadata, etc.)</td>
</tr>
</tbody>
</table>

3.6.1 Database User/Password Storage Locations

Database user and password information is stored in two locations:

1. ICP Service (Bifrost)
   a. File: `<InstallDir>\conf\services.properties`
   b. This file holds the database connection information that is used by the ICP service (Bifrost). An encrypted password is stored for the SYSTEM and ICP_P users, but the ICP_IS user normally uses a default password that is not specified in this file. If the default password is changed, then the information must be added to the file (detailed instructions follow).
JBoss (Wildfly)

a. File: `<InstallDir>\jboss\icpConfig\standalone\standalone.xml`
b. This file holds the database login information (ICP_P schema) for the JBoss application server. Follow instructions in section 2: JBoss Configuration Procedures.

3.6.2 Password Encryption

Bifrost and JBoss use different password encryption schemes.

- To change a password for Bifrost, the encrypted password in the `services.properties` file is simply replaced by the new plain-text password preceded by an exclamation point (`!<new_password>`). As soon as the ICP service is restarted, the plain-text password is automatically replaced by the encrypted password.

- The encrypted password for JBoss must be generated using the `encjboss.bat` script that is located in the `<InstallDir>\bin` directory. This script calls a JBoss component to encrypt the specified plain-text password and return it as a text value that can be copied into the `standalone.xml` configuration file described in the storage locations above.

Recommendations: Database passwords can be changed only when the ICP services are shut down.

Configuration Example: Change the password for the ICP_P user in the database server.

Configuration File Location: Oracle Database Server

Connect to the database as the SYSTEM user. The example shows how to connect to a local Oracle database using SQLplus.

```
sqlplus system/<password>@localhost/ICP
```

Change the user password. (Note that only the password itself is case sensitive.)

```
ALTER USER icp_p IDENTIFIED BY new_password;
```

Configuration Example: Change the database password for the ICP_P user for Bifrost. Search for `db.user.schema` entries in the `services.properties` file. Replace the encrypted password with a plain-text password preceded by an exclamation point (`!`).

Configuration File Location

```
<InstallDir>\conf\services.properties
```
The passwords are specific to a schema. The default system schema for Oracle is 'system' while the default schema for SQL Server is 'sa'. Thus to set the system Oracle password, use the following:

```
#db.user.schema.system=!password
#db.user.schema.sa=!password
```

Where the password for the default system schema in SQL Server would be set as follows:

```
db.definition.@sysadmin.schema=ICP_P
#db.user.schema.@sysadmin=VHxSUy81nBFAJU5z9BHoqZED95Wh1yD7Hc=
db.user.schema.@sysadmin=!new_password

db.definition.@product.schema=ICP_P
#db.user.schema.@product=R4evDc5u88WkGHVROTOZo8UWi2T5NHSC=
db.user.schema.@product=!new_password

db.definition.@ingenixsys.schema=ICP_IS
#db.user.schema.@ingenix_system=!new_password
```

**Configuration Example:** Change the database password for the ICP_P user for the JBoss application server. Search for `db.user.schema` entries in the `services.properties` file. Replace the encrypted password with a plain-text password preceded by an exclamation point (!).

**Configuration File Location**

```
<InstallDir>\jboss\icpConfig\standalone.xml\standalone.xml
```

1. Open a Command Prompt window, navigate to `<InstallDir>\bin`, and generate the new password:

   ```
   encjboss new_passwd
   Encoded password: 190086d21f07a929dd5e67b9d75dab4e
   ```

2. Open `standalone.xml` file (described earlier) with a text editor. Update the password and save the file.

   ```
   <security-domain name="encrypted-ds" cache-type="default">
   <authentication>
   <login-module code="org.picketbox.datasource.security.SecureIdentityLoginModule" flag="required">
   <module-option name="username" value="icp_p"/>
   <module-option name="password" value="190086d21f07a929dd5e67b9d75dab4e"/>
   <module-option name="managedConnectionFactoryName" value="jboss.jca:service=XATxCM,name=apollo-datasource"/>
   </login-module>
   </authentication>
   </security-domain>
   ```

3. Follow the instructions in section 2: JBoss Configuration Procedures to update your server.
3.7 Changing the Keystore/Truststore Passwords for HTTPS

**Background:** When SSL certificates are used to secure communication, they are stored in a Keystore file that is password protected. A default password is set by the base installer, but for added security, clients are encouraged to change the default password after CM/CES has been installed. In addition, good security practice dictates that the password be changed on a periodic basis.

Because the password is recorded in multiple configuration files, this section provides a short list of the files that must be updated as well as references to other sections in this document that describe the process for each file in more detail.

| Note | When the claim interface is also secured by SSL certificates, a Truststore file is also required which contains public keys. This file is also password protected and this password is specified in the same files that contain the Keystore password. Therefore, the file list below applies equally to changing both the Keystore and Truststore passwords. |

<table>
<thead>
<tr>
<th>File</th>
<th>Suggested</th>
</tr>
</thead>
<tbody>
<tr>
<td>standalone.xml</td>
<td>5.5.1 - Configuring the Standalone.xml File (see page 71)</td>
</tr>
<tr>
<td>jboss-ejb-client.properties</td>
<td>5.5.2 - Securing Remote EJB Access (jboss-ejb-client.properties)</td>
</tr>
<tr>
<td>Batch scripts</td>
<td>5.5.2.1 - Updating the Keystore Password in Batch Scripts</td>
</tr>
<tr>
<td>services.properties</td>
<td>5.5.3.1 - Configuring the services.properties File</td>
</tr>
<tr>
<td>icp.wrapper.conf</td>
<td>5.5.3.2 - Configuring the icp.wrapper.conf File</td>
</tr>
<tr>
<td>connector.wrapper.conf</td>
<td>5.5.4.1 - Configuring the connector.wrapper.conf File</td>
</tr>
<tr>
<td>broker.xml</td>
<td>5.5.4.2 - Configuring the broker.xml File (see page 78)</td>
</tr>
</tbody>
</table>

3.8 JBoss Logging Verbosity

**Background:** The JBoss application server logs messages to the jboss.server.log file in the `<InstallDir>\log`. The default logging level is WARN, meaning that only messages indicating a potential problem or an actual error get logged. To track claim processing in the logs, this level of logging is not sufficient. By increasing the log-level to INFO on specific Java classes (objects), useful information can be added to the logs without flooding the logs with unimportant messages or causing the log files to roll over too rapidly.

**Recommendations:** Log messages that track claim flow events (key steps in the analysis of a claim) and record the time required to perform analysis should be enabled when there is any concern about claim processing or analysis performance.

**Configuration Example:** Change the log-level (priority) for the following Java classes from WARN to INF.
Note

In previous versions of CM/CES that were based on the JBoss 4 application server, changes to logging configuration were automatically detected by JBoss and a service restart was not required. However, due to the upgrade to Wildfly, this is no longer the case since it maintains tighter control over the configuration file and expects changes to be made via either the JBoss console or a command-line configuration tool. Therefore, after updating the master file in the `icpConfig` folder, services must be stopped before following the instructions in section 2: JBoss Configuration Procedures.

Configuration File Location

`<InstallDir>\jboss\icpConfig\standalone.xml\standalone.xml`

```xml
<logger category="com.ingenix.freya.messaging.ejb.ClaimFlowMDB">
  <level name="INFO"/>
</logger>

Sample Log Message

2014-06-16 14:47:24,203 INFO (Thread-41 (group:HornetQ-client-global-threads-2089315596))

<logger category="com.ingenix.freya.messaging.ResultRouterImpl">
  <level name="INFO"/>
</logger>

Sample Log Message

[com.ingenix.freya.messaging.ResultRouterImpl] CLAIM FLOW EVENT (Send Results): MsgID:[607043c5-7362-4b2a-8140-8719ba235e02] ClaimID:[50608592] originally from [Real Time Port 10050]

<logger category="com.ingenix.freya.claimedit.ejb.ClaimLineGroup">
  <level name="INFO"/>
</logger>

Sample Log Messages

2014-06-16 14:47:25,543 INFO (Thread-41 (group:HornetQ-client-global-threads-2089315596))

3.9 Claim History Retrieval

Note: This section is not applicable for users whose host system sends all patient history with each claim (most payer installations).

Background: Before a claim is analyzed, all previous claims for the same patient (claim history) must be retrieved from the database so they can be passed to ILOG to execute rules that include history data as part of the analysis. To improve database access performance, the ability to retrieve all claim data (claim body, claim lines, modifier code, flags, etc.) in a single query that joined data from multiple tables has been added. The previous method was to perform separate queries against each of the tables that stored claim detail (claim lines, modifiers, flags, etc.). Both the old and new claim history methods are available in the software.

At one point, the new “single query” method was enabled by default, but this has since been changed so that the original “multi-query” method is now the default. This is because testing showed that a flaw in the method made performance much worse when there are larger numbers of historical claims for a patient.

Recommendations: Due to a known issue with the “single-query” claim history retrieval method, it is recommended that the “multi-query” method of retrieving claim history be used.

Configuration Example: After logging into the database as the icp_p user with a SQL query editor (SQLplus, etc.), execute the following SQL statement:

```sql
Database: CONFIGPROPERTY table

-- Oracle
insert into CONFIGPROPERTY(CONFIG_PROPERTY_IID, PATH, PROPERTY_NAME, PROPERTY_VALUE )
values (S_CONFIG_PROPERTY.nextval, '/ClaimServiceSettings', 'HistoryFetchMethod', '2');
commit;

-- SQL Server
insert into CONFIGPROPERTY(PATH, PROPERTY_NAME, PROPERTY_VALUE )
values ('/ClaimServiceSettings', 'HistoryFetchMethod', '2');
go
```
...or simply delete the property if it exists.

```sql
-- Oracle and SQL Server
delete from CONFIG_PROPERTY
where PATH='/ClaimServiceSettings' and PROPERTY_NAME='HistoryFetchMethod';
```

To use the “multi-query” method of retrieving claim history, update the property and set the property value to “1”...

```sql
-- Oracle and SQL Server
update CONFIG_PROPERTY set PROPERTY_VALUE='1'
where PATH='/ClaimServiceSettings' and PROPERTY_NAME='HistoryFetchMethod';
```

Remember to commit the transaction if your SQL editor does not have auto-commit enabled.

### 3.10 Cross Claim Type Editing

**Background:** The "Cross Claim Type Editing" feature is related to patient claim history retrieval. The objective is to include Facility patient history claims when editing a Professional claim, and to include Professional patient history claims when editing a Facility claim. This provides a more complete patient history when analyzing the current claim.

Normally, patient history claims are only retrieved from the same enterprise as the current claim. Professional and Facility claims are associated with different enterprises. To retrieve “cross claim type” history claims the Professional enterprise needs to be linked to a corresponding Facility enterprise and vice versa. This is configured through the System Settings → Enterprise Group Configuration menu option.

Enabling/disabling the Cross Claim Type editing feature must be done via database queries that add or modify two system configuration properties (see instructions below). When Cross Claim Type Editing is enabled, “Enterprise Groups” must also be configured as explained above.

**Recommendations:** Cross claim type editing should be enabled only when absolutely required. Turning this feature on may have significant performance impact.

**Configuration Example:** Connect to the database as the icp_p user with a SQL query editor (SQLplus, etc.) and execute the SQL statements provided below.

```
Note
“Cross Claim Type” editing is assumed to be disabled if the CrossClaimEditing configuration property does not exist.
```

**Configuration File Location**

**Database:** CONFIGPROPERTY table

First check whether or not each of the required properties already exists.

```sql
-- CHECK IF PROPERTY EXISTS (Oracle or SQL Server)
select * from CONFIG_PROPERTY
where PATH = 'icp/claim' and PROPERTY_NAME = 'CrossClaimEditing';

select * from CONFIG_PROPERTY
where PATH = '/icp/claimedit' and PROPERTY_NAME = 'history.include.enterprisegroup';
```
If either of the above queries do not return a result, insert the required property.

```
-- INSERT PROPERTY IF NOT PRESENT (Oracle)
insert into CONFIG_PROPERTY(CONFIG_PROPERTY_IID, PATH, PROPERTY_NAME, PROPERTY_VALUE)
values (S_CONFIG_PROPERTY.nextval, 'icp/claim', 'CrossClaimEditing', 'Y');
insert into CONFIG_PROPERTY(CONFIG_PROPERTY_IID, PATH, PROPERTY_NAME, PROPERTY_VALUE)
values (S_CONFIG_PROPERTY.nextval, '/icp/claimedit', 'history.include.enterprisegroup', 'Y');
commit;

-- INSERT PROPERTY IF NOT PRESENT (SQL Server)
insert into CONFIG_PROPERTY(PATH, PROPERTY_NAME, PROPERTY_VALUE)
values ('icp/claim', 'CrossClaimEditing', 'Y');
insert into CONFIG_PROPERTY(PATH, PROPERTY_NAME, PROPERTY_VALUE)
values ('/icp/claimedit', 'history.include.enterprisegroup', 'Y');
go
```

If the property is found by the above query, update the property.

```
-- UPDATE EXISTING PROPERTY (Oracle)
update CONFIG_PROPERTY set PROPERTY_VALUE = 'Y'
where PATH = 'icp/claim' and PROPERTY_NAME = 'CrossClaimEditing';
update CONFIG_PROPERTY set PROPERTY_VALUE = 'Y'
where PATH = '/icp/claimedit' and PROPERTY_NAME = 'history.include.enterprisegroup';
commit;

-- UPDATE EXISTING PROPERTY (SQL Server)
update CONFIG_PROPERTY set PROPERTY_VALUE = 'Y'
where PATH = 'icp/claim' and PROPERTY_NAME = 'CrossClaimEditing';
update CONFIG_PROPERTY set PROPERTY_VALUE = 'Y'
where PATH = '/icp/claimedit' and PROPERTY_NAME = 'history.include.enterprisegroup';
go
```

3.11 Claim Line Expansion

The “Claim Line Expansion” feature divides a single claim line into multiple lines if the date of service range matches the units (for example, the date range spans a period of five (5) days and there are five (5) units on the line). Each line corresponds to one unit per day and is edited separately. Claim Line Expansion is used to perform partial denials of individual claim lines. When the “Claim Line Expansion” feature is disabled, CM/CES will still apply edits appropriately to the claim line.

When the feature is enabled, you may also configure the maximum number of units that will be expanded. The reason for this is that if a single line expands into a very large number of lines (for example, 500 or more), it will take much longer to process the claim because each line is edited separately. To prevent this from occurring and causing claim timeouts, the maximum number of units that will be expanded can be specified by the max.units.for.line.expansion configuration property.

EXAMPLES:

1. The claim line has a date range spanning five (5) days, with five (5) units. The line is expanded with each line having a different day within the date range and one unit. Each line is edited separately and can be flagged separately.

2. The claim line has a date range spanning five (5) days, with four (4) units. The line is not expanded.

3. The max.units.for.line.expansion property is set to 100 and the claim line has a date range of one year (365 days) with 365 units. The line is not expanded because the maximum allowed lines would be exceeded.
Background: The “Claim Line Expansion” feature is enabled or disabled via an entry in the CONFIG_PROPERTY database table. The default value of this property was false (disabled) prior to 5.4 CU03 if the configuration property did not exist. 5.4 CU03 adds the configuration property with a value of true (enabled) for all systems where the property did not previously exist.

To disable line expansion, the expand.lines property must be modified in the CONFIG_PROPERTY database table.

Note: As of 5.4 SP2-CU02, a new max.units.for.line.expansion property is added to the CONFIG_PROPERTY table with a value of 500. The property value can be modified to a value that is appropriate for your system to ensure that this feature does not cause claim timeouts. If 5.4 SP2-CU02 or later has not been installed, adding this property will have no effect.

Recommendations: Enabling this feature may impact performance, especially when processing claim lines with higher unit values. Each user must determine the proper setting of the expand.lines property for their system. The max.units.for.line.expansion property can also be set to limit the number of lines that can be generated to prevent claim timeouts that would otherwise occur when this feature is enabled.

Configuration Example: Connect to the database as the icp_p user with a SQL query editor (SQLplus, etc.) and execute the SQL statement provided below to disable the “Claim Line Expansion” feature. (Set the property value to true to enable the feature.) You may also change the value of the max.units.for.line.expansion property as needed for your system.

Configuration File Location

Database: CONFIG_PROPERTY table

--- Oracle
Update CONFIG_PROPERTY set PROPERTY_VALUE = 'false'
  where PATH = '/icp/claimedit' and PROPERTY_NAME = 'expand.lines';
Update CONFIG_PROPERTY set PROPERTY_VALUE = '500'
  where PATH = '/icp/claimedit' and PROPERTY_NAME = 'max.units.for.line.expansion';
commit;

--- SQL Server
Update CONFIG_PROPERTY set PROPERTY_VALUE = 'false'
  where PATH = '/icp/claimedit' and PROPERTY_NAME = 'expand.lines';
Update CONFIG_PROPERTY set PROPERTY_VALUE = '500'
  where PATH = '/icp/claimedit' and PROPERTY_NAME = 'max.units.for.line.expansion';
go

3.12 Force New/Updated DDR to Test/Live on KB Load

Background: When a new KnowledgeBase (KB) is loaded on a periodic basis, it will include Data-Driven Rule updates, some of which will have a status of “New” or “Updated”. Normally, these rules require manual review and the user must change their status to “Test” or “Live” before can be used for claim analysis. A new option to automatically set all DDR with a status of “New” or “Updated” to either “Test” or “Live” status (eliminating the need for manual operations) is now available via a SQL script that can be run to add the configuration to the DDR_PROPERTIES table.

Prerequisites: July 2017 KB: CES/CM_KB_2017_Q3A_5.0-5.4.
**Recommendations:** The setting of this parameter is up to the client. Some users will not want to use this option because every rule must typically go through extensive evaluation and testing before it is ever implemented in production.

**Configuration Example:** After logging into the database as the icp_p user with a SQL query editor (SQLplus, etc.), execute the following SQL statement:

1. **Auto-set New/Updated KB rules to Test status.**

   ```sql
   insert into DDR_PROPERTIES (PROP_TYPE,PROP_NAME,VALUE) 
   values ('RULE_STATUS','autosetonload','TEST');
   commit;
   ```

2. **After initial auto-set of Test, change rules to Live status.**

   ```sql
   Update DDR_PROPERTIES set VALUE='LIVE' where PROP_TYPE='RULE_STATUS' and PROP_NAME='autosetonload';
   commit;
   ```

3. **Remove the Auto-set New/Updated KB rules function.**

   ```sql
   delete from DDR_PROPERTIES where PROP_TYPE='RULE_STATUS' and PROP_NAME='autosetonload';
   commit;
   ```

### 3.13 Claim Connector Configuration

There are several configuration options available for the **Claim Connector** specified via the service wrapper that controls the **ICP_Connector** service. Because each type of system (client) that sends claims to the **Claim Connector** is unique, it is important to set the configuration options to best match the characteristics of the client.

There are two basic types of clients:

1. **Asynchronous.** These clients make a single TCP connection and send multiple claims via the connection without waiting for a response. The claim responses are received asynchronously over the same TCP connection and may be received in a different order from which they were sent due to the multi-threaded claim analysis and varying times required to complete the analysis. The client should have configuration to control the number of “in flight” claims (i.e., the Optum `batchfile.exe` client).

2. **Synchronous.** These clients make one TCP connection for each claim and send a single claim. The client process or thread waits for a configured timeout period for the claim response to be received, so the connection is closed when either the claim response is received or the response timeout expires. The client can also be configured as to the number of concurrent sessions that are allowed; this controls the maximum number of “in flight” claims.

| Note | The RESTful connection type is by definition a synchronous, multi-session interface due to the nature of the HTTP protocol upon which it is based: It sends a “request” for the analysis of a single claim, then waits for a “response,” at which point the connection is closed. |
3.13.1 Configuration for Multi-Session and Single-Session Clients

**Background:** Multi-session clients frequently open and close TCP connections because there is one connection per claim. Each time a TCP connection is opened, the Claim Connector must create a message handler to receive the message, translate the plain text claim payload into a Java “Claim” object, and send the Java claim in a JMS message to the JBoss application server for analysis.

The message handler has a JMS producer (sender) and consumer (receiver) for sending and receiving JMS messages. Because creating these components is time consuming, message handlers are pooled so they can be reused and thus eliminate the overhead of object creation each time a new TCP connection is made. The following provides more technical detail regarding the message handler pool.

- When a client disconnects, its JMS resources are not released but are held with the message handler in a handler pool for use by future connections. If a client attempts to connect to the Claim Connector and no handler is available in the handler pool, one is created and added to the pool.

- When the handler pool reaches its configured maximum size, new client connections are refused until an active handler is returned to the pool (i.e., a claim response is received by another handler, its connection is closed and the handler is put back into the pool). During that time, communication is suspended and it may appear that the Claim Connector has stopped working. For this reason, it is critical that the maximum number of connections configured for the client is well understood so that the maxPoolSize parameter for the Claim Connector can be configured to support at least the same number of connections plus a small buffer.

- The preloadConnectionPool parameter is designed to alleviate another situation where the client immediately makes a large number of connections to the Claim Connector upon startup. Because the creation of each handler is time consuming (especially the creation of the JMS producer and consumer), slow handler creation can cause connection timeouts or connection refusals. This can be resolved by setting the preloadConnectionPool parameter to true so that when the Claim Connector starts up, the handler pool is “prefilled” with handlers of the specified maxPoolSize parameter value or 100, whichever is less.

**Recommendations:**

- For asynchronous clients where a single TCP connection is made, the prefillConnectionPool parameter should be set to false.

- For synchronous clients where many concurrent TCP connections are required, the prefillConnectionPool parameter should be set to true. The default value of 256 for the maxPoolSize parameter should be sufficient for most installations; however, this should be confirmed by the customer, who should verify the maximum number of concurrent connections allowed by the client. There should be a comfortable margin above the largest number of expected concurrent client connections. There is no limit to how high this can be set. However, if the maxPoolSize parameter is set to a very large value (i.e., over 1500), the maximum memory allocation for the Claim Connector’s JVM may also need to be increased in connector.wrapper.conf.
**Configuration Example:** Set the `maxPoolSize` property to the estimated limit plus margin and set the `preloadConnectionPool` property to `true`.

**Configuration File Location**

```
<InstallDir>\conf\connector.wrapper.conf
```

```
# Java Additional Parameters
wrapper.java.additional.1=-Xms512m
wrapper.java.additional.2=-Xmx1024m
wrapper.java.additional.3=-XX:+UseG1GC
:
wrapper.java.additional.14=-Djava.net.preferIPv4Stack=true
wrapper.java.additional.15=-Dicp.connector.maxPoolSize=256
wrapper.java.additional.16=-Dicp.connector.preloadConnectionPool=true
:
```

Add around 200 to this value if pool size greater than 1500.

Set value to the maximum expected number of concurrent connections.

### 3.13.2 Prevention of Parallel Patient Claim Processing (pre-SP1 CU04)

**Note**

Significant changes were made to the function described in this section with the release of 5.4 SP1 CU04. If you have installed this Cumulative Update or a later one, please skip this section and refer to the following section instead.

**Background:** When claims are analyzed, some of the rules check “frequency” constraints. For example, a particular policy may only allow a subscriber to have one wellness checkup per year, or three dialysis treatments per week. If multiple claims for the same patient are processed at the same time, each claim will not yet be recorded in the database and thus will not yet be available as “history” claims for the other claims that are concurrently processing. They will therefore not be able to accurately perform these types of frequency checks.

It is important for this type of “parallel” or concurrent processing of claims for the same patient to be prevented in most cases. Most adjudication systems that interface with **Claims Edit System** send the patient history claims with the claim to be analyzed, which eliminates the need to retrieve the history from the database. Also, when installed in a clearinghouse environment, **Claims Manager** may not need to provide 100% accuracy for frequency edits because only a limited amount of patient history is stored or because the frequency rules are not included in the rulesets.

Because ensuring serial processing of claims for the same patient adds overhead and can reduce throughput by 10 to 20 percent (depending on the mix of claims in a particular batch), this feature can be disabled.

**Recommendations:** The prevention of parallel patient claim processing (PPP) feature should normally be enabled for **Claims Manager** clients and disabled for **Claims Edit System** clients when patient history is submitted with each claim. Also, when **Claims Manager** is deployed in a clearinghouse environment, it should normally be disabled for increased performance.
**Configuration Example:** Set the `icp.connector.parallelPatientProcessing` property to `hornetq` to prevent PPP and set the property to `off` to disable the feature.

### Configuration File Location

```
<InstallDir>\conf\connector.wrapper.conf
```

<table>
<thead>
<tr>
<th>Wrapper Java Additional Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wrapper.java.additional.1=-Xms512m</code></td>
</tr>
<tr>
<td><code>wrapper.java.additional.2=-Xmx1024m</code></td>
</tr>
<tr>
<td><code>wrapper.java.additional.3=-XX:+UseG1GC</code></td>
</tr>
<tr>
<td><code>wrapper.java.additional.14=-Djava.net.preferIPv4Stack=true</code></td>
</tr>
<tr>
<td><code>wrapper.java.additional.15=-Dicp.connector.maxPoolSize=256</code></td>
</tr>
<tr>
<td><code>wrapper.java.additional.16=-Dicp.connectorpreloadConnectionPool=true</code></td>
</tr>
<tr>
<td><code>wrapper.java.additional.17=-Dicp.connector.parallelPatientProcessing=hornetq</code></td>
</tr>
</tbody>
</table>

---

### 3.13.3 Patient Claim Serialization (post-SP1 CU04)

**Note** The function described in this section was first included with Cumulative Update 5.4 SP1 CU04. If you have not yet installed this cumulative update or a later one, please refer to the preceding section instead.

**Background:** When claims are analyzed, some of the rules require patient history for analysis, such as Frequency, Global and Unbundle editing. To ensure clinical accuracy, it is necessary for claims for the same patient to be processed serially rather than concurrently so that each claim has the full and correct set of historical claims.

If multiple claims for the same patient are processed at the same time, each claim will not yet be recorded in the database and thus will not yet be available as “history” claims for the other claims that are concurrently processing. They will therefore not be able to accurately perform these types of history checks.

**Patient Claim Serialization** is managed by a module in the Claim Connector. This module tracks each incoming claim via a separate queue for each unique patient MRN. During the time that a claim is being analyzed, any other claims that are received and which are for the same patient as the claim being analyzed are held in the queue for that patient until the claim response is received. This event releases the next waiting claim for that patient. When the claim response for the last claim that is queued for that patient is received, the queue is deleted. The diagram below shows this process.

![Diagram of Patient Claim Serialization in the Claim Connector](image)

**Figure 1 – Patient Claim Serialization in the Claim Connector**

© 2020 Optum, Inc.
In the diagram above, the patient claims at the head of each patient queue in the Patient Claim Serialization module correspond to the claims that are being analyzed in the JBoss Application Server. When the claim response is sent back to the Claim Connector, the claim at the head of the PCS patient queue is removed and the next claim in the queue for that patient is released and sent to JBoss for analysis (see P12 and P07 above). If the patient queue is empty when the head claim is removed (P01, P29, and P82), that patient queue is deleted. Note that the incoming claim for patient P41 will cause a new patient queue to be created by the PCS module when it is received.

**Note**

When a claim that has been waiting in a patient queue is released by a claim response, it is assigned HIGH priority when placed on the JMS ‘analyze’ queue. This means that it will “jump to the head of the line” relative to any other lower-priority messages that are already on the queue. This is to ensure that as little additional wait time as possible is incurred before analysis of the claim begins.

The Patient Claim Serialization module in the Claim Connector has several configuration options that are found in the connector.wrapper.conf file. These are explained below.

- **icp.connector.pcs.mode** - This parameter has three allowed values (which are not case sensitive):
  
  - **off** - Disables patient claim serialization. This option should be used when patient history claims are sent with the claim, which is the case for most CES clients.
  
  - **hornetq** - Patient Claim Serialization is managed by the Grouping Handler component of the JMS provider (previously HornetQ and now Artemis). It is not recommended that this option be used due to its “strict FIFO” message delivery rule which can significantly degrade performance.
  
  - **connector** - Patient Claim Serialization is managed within the Claim Connector as described in this section. This is the recommended setting.

- **icp.connector.pcs.clientTimeout** - It is very common for software clients that interface with the connector to have a timeout within which they expect CM/CES to return a claim response. If the claim response is not received within the timeout period, it is considered to be a failure to process the claim and error handling on the client side is performed. For synchronous clients, the client closes the TCP connection to allow another claim to be sent. This timeout value for the Claim Connector should be set to the same value that is used by the client software to decide when a claim response has timed out.

- **icp.connector.pcs.responseTimeout** - This is the amount of time that a claim is allowed to wait at the head of the patient queue in the PCS module. When the timeout expires, the claim is removed from the queue and the next claim waiting in the patient queue is released.
**Note** When a claim at the head of the patient queue times out due to the response timeout having expired (or due to a normal response having been received), each claim that was waiting behind it in the patient queue is checked to determine whether the client timeout has expired or not (see `clientTimeout` parameter above). If it has, that claim is removed from the patient queue and is not sent to JBoss for analysis. This is because it is assumed that the client that sent the claim has already timed out and closed the TCP connection; therefore, it would not be possible to send a claim response even if the processing were to be performed. Therefore, to eliminate unnecessary load on the system, the claim is discarded without performing clinical analysis and no attempt is made to send a response to the client.

- **icp.jms.maxQueueDepth** - This parameter is not fully implemented in the SP1 CU04 update, but will be in a later update.

**Recommendations:** The Patient Claim Serialization (PCS) feature should normally be enabled for Claims Manager clients and disabled for Claims Edit System clients when patient history is submitted with each claim. Also, when Claims Manager is deployed in a clearinghouse environment, it may be disabled for increased performance.

**Configuration Example:** Set the `icp.connector.pcs.mode` property to `connector` to enable PCS and set the property to `off` to disable the feature.

**Configuration File Location**

```
<InstallDir>\conf\connector.wrapper.conf
```

```java
wrapper.java.additional.1=-Xms512m
wrapper.java.additional.2=-Xmx1536m
wrapper.java.additional.3=-XX:+UseG1GC

wrapper.java.additional.11=-Djavax.net.ssl.trustStorePassword=ingenix1
wrapper.java.additional.12=-Djava.net.preferIPv4Stack=true
wrapper.java.additional.13=-Dicp.connector.maxPoolSize=50
wrapper.java.additional.14=-Dicp.connector.preloadConnectionPool=false
wrapper.java.additional.15=-Dicp.connector.pcs.mode=connector

# pcs.clientTimeout - Seconds the will client wait for a response before timing out.
wrapper.java.additional.16=-Dicp.connector.pcs.clientTimeout=300

# pcs.responseTimeout - Seconds the connector will wait for a response before
# timing out and sending the next queued claim.
wrapper.java.additional.17=-Dicp.connector.pcs.responseTimeout=300

# jms.maxQueueDepth - Message count in 'analyze' queue after which the
# Connector starts applying 'back pressure'.
wrapper.java.additional.18=-Dicp.jms.maxQueueDepth=300
```
3.13.4 Startup Timing

**Background:** The CM/CES application is comprised of three system services that interact with one another.

- **ICP Service** - Responsible for starting/stopping the other services and for data loading.
- **ICP Connector** - Receives claims via interfaces to external clients and distributes them to the processing engine via JMS (Java Messaging System). The Artemis JMS server is bundled with the Claim Connector (it runs in the same JVM).
- **ICP Engine** - The core application that contains the rules engines for claim analysis. This is a JEE application that runs in the Wildfly (JBoss) application server.

After the ICP Service is started, it next starts the ICP Connector service with the JMS server, after which the ICP Engine service is started. However, configuration for the Claim Connector is contained in the database and the Claim Connector does not have any connection to the database. Instead, it must wait for the ICP Engine to become available, then send a request for configuration information via a JMS message.

**Recommendations:** This setting does not typically need to be changed from the default value of -1. However, changing the value to 0 or greater can be helpful in order to make sure that no connections are opened to receive claims until after the ICP Engine service is fully initialized (including loading of data caches). This prevents premature attempts to process claims.

**Configuration File Location**

```
<InstallDir>\conf\connector.wrapper.conf
```

```
wrapper.java.additional.20=-Dicp.connector.startMessagingService=true
# icpReadyTimeout - Specifies whether or not the connector will wait for notification
# from JBOSS that ICP is ready before it sends claim messages.
# -1 : The connector will not wait for an ICP ready notification, it assumes that a
#       JMS server and JBoss instance are ready to provide connection information.
#  0 : The connector will wait indefinitely to receive an ICP ready notification before
#       it attempts to request connection properties.
#  >0 : The number of minutes the connector will wait to receive an ICP ready
#       notification. If it gets the notification before this time lapses, it will
#       request connection properties otherwise, it will shutdown.
wrapper.java.additional.21=-Dicp.connector.icpReadyTimeout=-1
# wrapper.java.additional.99=-Duser.timezone=US/Mountain
```

3.13.5 Handling Large Claim Messages

**Background:** Payers that use Claims Edit System typically send all patient history claims along with each current claim that is to be edited. When there are many history claims, this can result in a very large message (10 MB or more). For maximum performance, it is best to configure the system so that these large messages are kept in memory while being communicated rather than being sent in “chunks” that are persisted to disk by the JMS server.

The Artemis message broker supports the sending and receiving of large messages even when the client and server are running with limited memory. To send a large message, the Claim Connector sends the message body to the JMS server with an InputStream identified. The server opens the InputStream and receives the entire large message in a stream of fragments which the server persists to disk as it receives them. When the time comes to deliver them to a consumer (JBoss), the server initially sends just the message with an empty body, after which an OutputStream is opened through which the entire large message is streamed over the TCP.
connection without the need to hold it in memory. At no time is the entire message body stored fully in memory on either the client or the server. This process of opening and closing an output and input stream for each message with intermediate data persistence to disk is slower than passing the entire message through a memory queue. It is desirable to configure the system to allow the large majority of messages to be passed via the memory queue.

**Recommendations:** Any message larger than the size specified by the `minLargeMessageSize` parameter is considered a large message. This parameter should be set large enough to handle most claims with patient history so that only a small percentage of larger claims will be handled by the special “Large Message” mechanism described in the previous paragraph.

### Note
Apache ActiveMQ Artemis messages are encoded using 2 bytes per character so if the message data is filled with ASCII characters (which are 1 byte) the size of the resulting Apache ActiveMQ Artemis message would roughly double. This is important when calculating the size of a “large” message as it may appear to be less than the `minLargeMessageSize` before it is sent, but it then turns into a “large” message once it is encoded.

**Configuration Example:** Set the `minLargeMessageSize` property to a value (in bytes) that is large enough to hold around 90% of “claim plus history” messages in memory. Multiply the character count of a “large claim” by two as indicated in the note above. The default value below is 400 KB.

**Configuration File Location**

```
<InstallDir>/conf/jndi.properties
```

```
### HA and ssl ###
ConnectionFactory.ConnectionFactory=(
  tcp://<mainNameOrIP>:61617
  ?sslEnabled=true;enabledProtocols=TLSv1%2CTLSv1.1%2CTLSv1.2,
  tcp://<standbyNameOrIP>:61617
  ?sslEnabled=true;enabledProtocols=TLSv1%2CTLSv1.1%2CTLSv1.2
)
?consumerWindowSize=10240
&minLargeMessageSize=40960
&blockOnDurableSend=false
&preAcknowledge=true
&threadPoolMaxSize=-1
&ha=true
&retryInterval=1000
&retryIntervalMultiplier=1.0
&reconnectAttempts=-1
```

Set this value to approximately twice the size of the number of characters in a large claim with patient history.

One other configuration parameter that must be taken into consideration is the `<journal-buffer-size>` parameter in the `broker.xml` file. This parameter specifies the largest “chunk” of data that can be written to the message journal files and must be larger than the largest single message written to the file. Therefore, if the minimum large message size is increased, it allows larger messages to be sent on JMS queues and therefore requires that the `<journal-buffer-size>` parameter be set larger than the `minLargeMessageSize` parameter value.

In `broker.xml`, the `<journal-buffer-size>` parameter is commented out by default, so the size used is the default size of 490K. If the `minLargeMessageSize` parameter is set to a value larger
than 490K, then the `<journal-buffer-size>` parameter must be uncommented and set to a larger size as shown below.

<table>
<thead>
<tr>
<th>Configuration File Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;InstallDir&gt;</code>\conf\broker.xml</td>
</tr>
</tbody>
</table>

```

<!-- this could be ASYNCIO or NIO -->
<journal-type>NIO</journal-type>
<!-- Change this from the default of 490K only when min-large-message-size is greater -->
<journal-buffer-size>501760</journal-buffer-size>
<paging-directory>../artemis/data/paging</paging-directory>
/bindings-directory>../artemis/data/bindings</bindings-directory>
/journal-directory>../artemis/data/journal/master</journal-directory>
```

- Delete the comment characters highlighted in **YELLOW**.
- Change the value (in bytes) of the `<journal-buffer-size>` parameter to a value approximately 100KB larger than the value of the `minLargeMessageSize` parameter in the `jndi.properties` file.

### 3.14 User Authentication Using an LDAP Server

#### Prerequisites

Service Pack 1 – Cumulative Update 04 must be installed before configuring LDAP.

**Background:** Most organizations use an LDAP server such as Microsoft’s *Active Directory* or the open source *OpenLDAP* server. LDAP allows many different applications and services to connect to the LDAP server to validate users. Microsoft implements the LDAP functionality within dedicated *Domain Controllers* in a domain-based environment to integrate security and object management.

Note that abbreviations for LDAP terminology are commonly used when configuring LDAP. The table below explains the meaning of each of the abbreviations.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dn</td>
<td>Distinguished name - This refers to the name that uniquely identifies an entry in the directory.</td>
</tr>
<tr>
<td>dc</td>
<td>Domain component - This refers to each component of the domain. For example, <code>www.google.com</code> would be written as <code>dc=www,dc=google,dc=com</code>.</td>
</tr>
<tr>
<td>ou</td>
<td>Organizational unit - This refers to the organizational unit (or sometimes the user group) that the user is part of. If the user is part of more than one group, you may specify as such (e.g., <code>ou=BeanCounters,ou=Accounting</code>).</td>
</tr>
<tr>
<td>cn</td>
<td>Common name - This refers to the individual object (person's name, meeting room, recipe name, job title, etc.) for whom/which you are querying the LDAP system.</td>
</tr>
<tr>
<td>mail</td>
<td>Mail – This refers to e-mail addresses or aliases which may also be used for security.</td>
</tr>
<tr>
<td>sn</td>
<td>Surname - This refers to a person’s surname.</td>
</tr>
</tbody>
</table>
A server holds a subtree starting from a specific entry (e.g., \texttt{dc=example,dc=com}) and may include its sub-domains or its children. Servers may also hold references to other servers, so an attempt to access \texttt{ou=accounting,ou=finance,dc=example,dc=com} could return a referral or continuation reference to a server that holds that part of the directory tree. Some servers also support chaining, which means the server contacts the other server and returns the results to the client.

LDAP rarely defines any ordering. The server may return the values of an attribute, the attributes in an entry, and the entries found by a search operation in any order. This follows from the formal definitions - an entry is defined as a set of attributes, and an attribute is a set of values. Sets need not be ordered.

**User Configuration:** To authenticate a user via LDAP, the user must have an account in CM/CES. When adding the user, the “LDAP” authentication type must be specified for the user. The advantages of using LDAP for authentication include:

- If a user is terminated or leaves the company, there is only one place where the account needs to be deactivated. The CM/CES application will still have the user account, but it will not be possible to log in with that user name because it will no longer be authenticated by the LDAP server.

- Users can use their corporate login ID (username) and password. When they change their password in the corporate system, it is also changed for logging into CM/CES.

**Recommendations:** Integration with an LDAP server is optional. Also, even if CM/CES is configured to allow user authentication via LDAP, it can still be specified individually for each user account.

**Configuration Example:** After stopping CM/CES services perform the following:

1. Navigate to the \texttt{<InstallDir>/jboss/icpConfig/standalone} directory and open the \texttt{standalone.xml} file with a text editor.

2. Search for \texttt{name="idp"}.

3. Insert the new \texttt{<login-module ...>} block (text inside the red outline box) as shown below.

   | Note | If the client has primary and secondary LDAP servers, they can both be included in the configuration by repeating the \texttt{<login-module>} element shown below and configuring the two instances for each of the two LDAP servers. However, a unique “name” attribute should be assigned to each of the two modules. (e.g., \texttt{name="LDAP_Server1"} and \texttt{name="LDAP_Server2"}). |

4. Add \texttt{cache-type="default"} to the \texttt{<security-domain>} element as shown below.

5. Modify the \texttt{FreyaDbLoginModule} login module entry and change the value of the \texttt{flag} attribute from \texttt{required} to \texttt{sufficient}.

6. Replace the text highlighted in \texttt{YELLOW} below with the hostname of the LDAP server and the port that it accepts on which it accepts requests. (e.g., \texttt{ldapsrv01.mycompany.com:389}).

7. Save the file and exit. (Be sure to follow section \texttt{JBoss Configuration Procedures}.)
Configuration File Location

<InstallDir>/jboss/ibpConfig/standalone/standalone.xml

: 
<security-domain name="idp" cache-type="default">
  <authentication>
    <login-module name="LDAP_Server" code="com.ingenix.sso.picketboxext.ICPLDAPLoginModule" flag="sufficient">
      <module-option name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory"/>
      <module-option name="java.naming.provider.url" value="ldap://<ldap.hostname>:<ldap.port>"/>
      <module-option name="java.naming.security.authentication" value="simple"/>
      <module-option name="bindDN" value="<ldap.bind.dn>"/>
      <module-option name="bindCredential" value="<ldap.bind.credential>"/>
      <module-option name="baseCtxDN" value="<ldap.context>"/>
      <module-option name="baseFilter" value="(<ldap.base.filter>={0})"/>
    </login-module>

    <login-module name="FreyaDbLoginModule" code="com.ingenix.sso.picketboxext.FreyaDbLoginModule" flag="sufficient">
      : 
    </authentication>
  : 
</security-domain>

Configuration Tips: The entries highlighted in YELLOW in the LDAP login module shown above must be customized for each client site to match the LDAP server configuration. The table below provides configuration tips for each of these values.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>LDAP Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.provider.url</td>
<td>&lt;ldap.hostname&gt;</td>
<td>The hostname and port specify how to contact the LDAP server or Domain Controller. “localhost” should be replaced with the Fully Qualified Domain Name (e.g., west.example.com) of the server or with its IP Address. Port 389 is the normal port for an Active Directory system or port 636 for secure (encrypted) connections. Apache LDAP servers normally use port 10389 or port 10636 for secure (encrypted) connections. Notes: 1. It is also possible to omit the hostname and specify just the domain for this property. This may help resolve some lookup issues. (e.g., use “ldap://company.com:389” instead of “ldap://host.company.com:389”).</td>
</tr>
<tr>
<td></td>
<td>&lt;ldap.port&gt;</td>
<td></td>
</tr>
<tr>
<td>bindDN</td>
<td>&lt;ldap.bind.dn&gt;</td>
<td>A bind DN is an object (distinguished name) within the LDAP directory which you bind to, inside LDAP, to give you permissions to do whatever you're trying to do. Some LDAP instances don't allow anonymous binds, or don't allow certain operations to be conducted with anonymous binds, so you must specify a bindDN to obtain an identity to perform that operation. Example: uid=admin,ou=administrators,ou=domainadmins,ou=security,dc=example,dc=com</td>
</tr>
<tr>
<td>Option Name</td>
<td>LDAP Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bindCredential</td>
<td><code>&lt;ldap.bind.credential&gt;</code></td>
<td>This is the password associated with the bindDN option. This cannot be an encrypted password, but there is an option for reading the password from another file so that the plain text password is no longer contained in the standalone.xml file, which does improve security, especially when there is a need to email the standalone.xml file to other parties for proofing or comment. For example <code>{EXT}type path_to_file_with_password</code>. An initial bind to the LDAP server is done using the bindDN and bindCredential options. The bindDN is some user with the ability to search both the baseCtxDN and rolesCtxDN trees for the user and roles. In some instances this credential is stored in a network share which has been locked down/secured to ensure only the application or Admin users have access.</td>
</tr>
<tr>
<td>baseCtxDN</td>
<td><code>&lt;ldap.context&gt;</code></td>
<td>This is the fixed distinguished name of the context to start the user search from. In other words, where the user’s security account is located within the LDAP tree or within Active Directory. The baseCtxDN path is constructed in Distinguished Name format. Example: <code>uid=admin,ou=administrators,ou=domainadmins,ou=security,dc=example,dc=com</code> Notes: 1. The path will most likely be unique for every client. What is important to remember is the CN location needs to be the OU where the referenced User Account resides. The rest of the path consisting of OU’s and DC’s needs to tree-down the remainder of the path in the exact order. (This can be viewed with an LDAP editor such as “ADSI Edit”.) 2. There are cases where the username lookup fails even though all values appear to be configured correctly. It has been found that removing the “ou” entries can resolve the problem (meaning “scan all organization units within the specified domain”). 3. Make sure that “ou” and “dc” values do not contain any spaces or dollar signs. This will cause the lookup to fail in almost all cases.</td>
</tr>
<tr>
<td>baseFilter</td>
<td><code>&lt;ldap.base.filter&gt;</code></td>
<td>This is the search filter used to locate the context of the user to authenticate. The input username/userDN as obtained from the login module callback will be substituted into the filter anywhere a “{0}” expression is seen. Examples: <code>uid</code> <code>sAMAccountName</code> (for Active Directory)</td>
</tr>
</tbody>
</table>

**Active Directory Configuration Example:** The example below can be modeled when integrating with Microsoft Active Directory (the most common LDAP server).

The **BLUE** entries below show how to configure the fully qualified hostname for the LDAP server.

The **GREEN** entries show how to configure the Account/Password for accessing the LDAP server in order to lookup users for authentication.

The **YELLOW** entries show how to configure the OU (organizational unit) within LDAP where CM/CES user accounts are located and the name of the variable (sAMAccountName) that will be set to the username to be authenticated.
Multiple LDAP Servers: It is common for an organization to have redundant LDAP servers for high availability. In order to make use of this capability in CM/CES, simply create two `<login-module>` entries like the one shown above; one for the primary LDAP server and one for the backup LDAP server. In the topmost entry, use the primary hostname and in the next entry use the backup hostname. If the primary LDAP server cannot be reached, the authentication process will automatically try reaching the backup LDAP server.

3.15 Reduce Claim Purge DB Blocking

Note: This section only applies to CM/CES systems that have installed 5.4 SP2-CU02 or later. Also, DB Blocking has mainly been seen in systems that use MS SQL Server, so this document is focused mainly on that database although it can apply to Oracle as well.

Background: When claims are purged at the same time that new claims are being processed, it is possible for there to be some database contention or “Blocking” between the two operations. To reduce the contention, claims are purged in batches of a relatively small size so commits can occur frequently and the duration of locks will be short. However, in spite of this strategy, some clients may still experience excessive blocking when both claim analysis and claim purging occur at the same time.

A message similar to the following will appear in the SQL Server logs when excessive blocking occurs:

```
5/6/2019 7:54:01 AM, Blocking Session Wait Time (Seconds) on ICPDB_PR is Warning.
SQL Server instance ICPDB_PR - Session 540 has been blocking other sessions for 25 seconds, since 5/6/2019 11:52:08 AM (UTC).
The session is being run on database [ICP] with application name 'SQLAgent - TSQL JobStep (Job 0x380c1a0d61a6458a7979f046646f52 : Step 1)' by user 'icp_p' on host 'ICPDB_PR'.
Last command issued at 5/6/2019 11:52:08 AM (UTC):
EXEC CLAIM_PURGE @CONTEXT = 1, @CLAIM_STATUS = 'I', @FREQUENCY = 0, @P_INCLUDE_NN_MFX = 0, @POLDER_THAN_DAYS = 120, @P_SCHEDULED_PURGE_IID = 65, @NPT = 0, @P_INCLUDE_NN_NPT = 0, @P_fe_module = 'Y', @P_fe_module = 'All', @P_ALL_ENTERPRISES = '1', @P_ALL_RULESETS = '1', @USER_ID = 1, @USERNAME = 'System'
```
Recommendations: Add a configuration parameter to the CONFIG_PROPERTY table to change the purge batch size from its default value. (For SQL Server, the default batch size is 2000. For Oracle, the default batch size is 10000.) Try using a value that is three quarters to one half of the default size. This will cause the purge to run somewhat slower, but it should improve reliability.

Configuration Example: This parameter is best set by using an SQL query editor such as SQL Server Management Studio.

Note
There are two versions of this feature. Be sure to use the correct set of queries below based on the CU level of your system.

1. The initial version was released with 5.4 SP2-CU02 and is configured via the DB_OPTIONS table.
2. A new version was released in 5.4 SP2-CU03 which is configured via the CONFIG_PROPERTY table where most other configuration properties are stored.

Configuration File Location (5.4 SP2-CU02)

Database: DB_OPTIONS table

Add the property using the following SQL query.

```sql
-- Oracle
insert into DB_OPTIONS(MODULE, OBJECTNAME, PARAMETER, PARAMETERVALUE)
values ('Purge', 'USP_CLAIM_PURGE', '@NUM_ROWS_TO_PURGE', '1500');
commit;

-- SQL Server
insert into dbo.DB_OPTIONS(MODULE, OBJECTNAME, PARAMETER, PARAMETERVALUE)
values ('Purge', 'USP_CLAIM_PURGE', '@NUM_ROWS_TO_PURGE', '1500');
go
```

Deleting the property will cause claim purge to revert to the default batch size.

```sql
-- Oracle and SQL Server
delete from DB_OPTIONS
where MODULE='USP_CLAIM_PURGE' and OBJECTNAME='@NUM_ROWS_TO_PURGE';
```

Use the following query to modify the parameter value after it has already been inserted.

```sql
-- Oracle and SQL Server
update DB_OPTIONS set PARAMETERVALUE='1000'
where MODULE='USP_CLAIM_PURGE' and OBJECTNAME='@NUM_ROWS_TO_PURGE';
```

Remember to commit the transaction if your SQL editor does not have auto-commit enabled.
Configuration File Location (5.4 SP2-CU03 or later)

Database: **CONFIG_PROPERTY** table

Add the property using the following SQL query.

```
-- Oracle
insert into CONFIG_PROPERTY(CONFIGPROPERTY_IID, PATH, PROPERTY_NAME, PROPERTY_VALUE)
values (S_CONFIG_PROPERTY.nextval, '/database/purge/USP_CLAIM_PURGE', 'NUM_ROWS_TO_PURGE', '1500');
commit;

-- SQL Server
insert into CONFIG_PROPERTY(PATH, PROPERTY_NAME, PROPERTY_VALUE)
values ('/database/purge/USP_CLAIM_PURGE', 'NUM_ROWS_TO_PURGE', '7500');
go
```

Deleting the property will cause claim purge to revert to the default batch size.

```
-- Oracle and SQL Server
delete from CONFIG_PROPERTY
where PATH='/database/purge/USP_CLAIM_PURGE' and PROPERTY_NAME='NUM_ROWS_TO_PURGE';
```

Use the following query to modify the parameter value after it has already been inserted.

```
-- Oracle and SQL Server
update CONFIG_PROPERTY set PROPERTY_VALUE='1000'
where PATH='/database/purge/USP_CLAIM_PURGE' and PROPERTY_NAME='NUM_ROWS_TO_PURGE';
```

Remember to commit the transaction if your SQL editor does not have auto-commit enabled.
4 Multi-Node Claim Processing with High Availability

To scale the processing capability of CM/CES software to match claim volume, a multi-node configuration may be necessary. As of the 5.4 Service Pack 1 release, CM/CES software no longer uses application server clustering; instead, all required communication between the several claim processing nodes is achieved via the Java Messaging Service. The specific JMS provider used is the Apache Artemis message broker that is now deployed in the Claim Connector rather than in each JBoss Application Server (Note that Apache Artemis is also the JMS provider that ships with Wildfly-10.). Only a single JMS server is active at a time whereas previously each JBoss instance had an active JMS server. However, backup Claim Connectors and associated JMS servers can be easily configured for failover to achieve high availability.

The diagram below shows the configuration of a typical multi-node CM/CES system for claims processing that is configured for High Availability.

![Multi-node System Configuration for High Availability](image)

The next diagram (Figure 3) shows essentially the same configuration, but from a “logical component” view that does not try to show the deployment on physical computers. Only one JMS server is active at a time. When it fails, the Claim Submitter switches to the first standby connector (this is not automatic in the product, but is usually managed by a load balancer of some kind) which has automatically detected that the main JMS server is no longer active and is now using its local JMS server. The JMS client in the JBoss application server does the same thing (auto-failover to a functioning standby JMS server). All of the dashed gray connections then become active. When the main Claim Connector and JMS server later become available again, the system will automatically failback to the main JMS server. The Claim Submitter can also be configured to return to submitting claims to the main Claim Connector.
The following sections describe in detail how to configure JMS communication for a multi-node CM/CES system with failover capability to provide High Availability. The configuration for each node is essentially the same except that nodes where the Claim Connector is running require extra configuration steps.

**Prerequisites**

- *Claims Manager* or *Claims Edit System* software has been identically installed on each node (computer) that will participate in the overall claim processing system. (Installation instructions are provided in separate documents: Claims Manager 5.4 Install Notes and CES 5.4 Install Notes.)
- One or more of the nodes must be designated to run the Claim Connector. The failover order (main to standby #1, standby #1 to standby #2, etc.) must be decided so all nodes can be configured identically.
- The ICP Service (Bifrost) is used on each node. However, it is configured to start the ICP Connector only on nodes where it should be running. The ICP Engine service (JBoss) must run on every node.
- All machines in the system must have network connectivity with each other as well as with the database server (*Oracle* or *Microsoft SQL Server*).
- All ICP services (*ICP Service*, *ICP Connector* and *ICP Engine*) must be stopped.

**Note**<InstallDir> in the following sections refers to the directory where the CM/CES software product is installed (e.g., C:\Optum\ICP, D:\Optum\ICP)

**4.1 Configuration for JMS Server Nodes**

High Availability is provided by running multiple JMS servers (message brokers) simultaneously on different ICP claim processing nodes. Although only one JMS server is active at a time, any one of them could take over the message traffic at any time should the currently active JMS server fail (usually caused by computer hardware failure). Typically, there is not a great advantage to having more than two JMS servers operational, but more than two can be configured if desired. Also note that the JMS server is embedded in the Claim Connector, so having a running JMS server implies that the Claim Connector is also running.

© 2020 Optum, Inc.
The instructions below apply to CM/CES 5.4 SP1 environments where High Availability (HA) is desired.

“Non-HA” environments are also supported in CM/CES 5.4 SP1. Non-HA means a standby JMS server (or claim connector) is not used. Refer to section 4.4: Non-HA Configuration for instructions on setting up a non-HA, multi-node CM/CES 5.4 SP1 environment.

4.1.1.1 Modifying JMS Server Node Configuration Files

The first step is to decide which nodes the JMS server should run on and the desired failover sequence among the several JMS servers.

Once the configuration has been decided, make the following modifications to configuration files on each JMS server node. (The details are explained in the next subsections.)

broker.xml – This file is used to configure the Apache Artemis message broker (JMS server). There are only very minor differences between the nodes where the JMS server is running.

jndi.properties – This file is used to configure the JMS client in the Claim Connector and should be exactly the same on each node where JMS server is running.

Before editing each file, it is recommended that you make a backup copy of the file renaming it so it can be clearly identified, possibly with a date/time stamp. (e.g., jndi.properties.20170315_1020)

4.1.2 Configuring the broker.xml File

The broker.xml file is also located in the <InstallDir>/conf directory. Edit this file with a text editor to make the changes shown below.

1. Change the <persistence-enabled> element from false to true. This is required to allow failover from one JMS Server to a standby server. This is because journal files are replicated from the active JMS server to all standby JMS servers to handle the case where messages cannot be lost when a failover occurs.

   Original configuration
   
   `<core xmlns="urn:activemq:core">
   <name>ICP</name>
   <persistence-enabled>false</persistence-enabled>
   <security-enabled>false</security-enabled>
   `<core xmlns="urn:activemq:core">

   New configuration
   
   `<core xmlns="urn:activemq:core">
   <name>ICP</name>
   <persistence-enabled>true</persistence-enabled>
   <security-enabled>false</security-enabled>
   `<core xmlns="urn:activemq:core">

2. Uncomment the appropriate <ha-policy> element by removing the <!--, and --> characters (highlighted in YELLOW in the example below).
a. If this is the main node, uncomment the `<ha-policy>` element under the “HA Policy for Main node” comment line. This element includes the inner `<master>` element.

```xml
<!--
<ha-policy>
  <replication>
    <master>
      <check-for-live-server>true</check-for-live-server>
    </master>
  </replication>
</ha-policy>
-->```

b. If this is a standby node, uncomment the `<ha-policy>` element under the “HA Policy for Standby node” comment line. This element includes the inner `<slave>` element.

```xml
<!--
<ha-policy>
  <replication>
    <slave>
      <allow-failback>true</allow-failback>
    </slave>
  </replication>
</ha-policy>
-->```

Locate the `<connectors>` element that immediately follows the `<ha-policy>` elements.

Uncomment the appropriate section for your configuration. Like the `jndi.properties` file, this section contains configurations for four different options:

- Single connector node without SSL (default).
- High Availability without SSL.
- Single connector node with SSL enabled.
- High Availability with SSL enabled.

a. Uncomment the `<connector>` elements below the comment line for the appropriate option. Make sure that the `<connector>` elements for all other options are commented out with `<!--` characters at the beginning of the line and `-->` characters at the end of the line.

b. In the uncommented lines, make sure that the `<mainNameOrIP>` and `<standbyNameOrIP>` placeholders are replaced with the actual hostname or IP address of the computer. The following example shows configuration for the “HA without SSL” option.

**Original configuration** (HA without SSL)

```xml
<!-- Single node without SSL settings (default) -->
<connector name="vm-connector">vm://0</connector>

<!-- HA without SSL settings template -->
<connector name="main-connector">tcp://MainHostname:61616</connector>
<connector name="standby-connector">tcp://StandbyHostname:61616</connector>
```
New configuration (HA with SSL)

```xml
<!-- Single node without SSL settings (default) -->
<connector name="vm-connector">vm://0</connector>

<!-- HA without SSL settings template -->
<connector name="main-connector">tcp://MainHostname:61616</connector>
<connector name="standby-connector">tcp://StandbyHostname:61616</connector>
```

**Note**
If more than one standby JMS server is needed, then add a `<connector>` element for each standby node and name them “standby1-connector”, “standby2-connector”, etc.

Locate the `<acceptors>` element that immediately follows the `<connectors>` element. This section has two configurations: one for non-SSL and one for SSL.

a. If you are enabling SSL, for either a single node or for multi-node high availability, comment out the two acceptors in the “Non-SSL” section (highlighted in YELLOW) and uncomment the acceptor in the “SSL settings template” section.

b. Also, if you have changed the default location for your `keystore` and `truststore` files, you must update the values highlighted in GREEN below.

```xml
<acceptors>
<!-- Non-SSL settings (default) -->
<!--acceptor name="netty-acceptor">tcp://0.0.0.0:61616 ?directDeliver=false</acceptor-->
<!--acceptor name="vm-acceptor">vm://0</acceptor-->

<!-- SSL settings template -->
<acceptor name="netty-acceptor">tcp://0.0.0.0:61617?directDeliver=true&sslEnabled=true;keyStorePath=C:/Optum/ICP/jboss/icpConfig/icp.keystore;keyStorePassword=ingenix1;trustStorePath=C:/Optum/ICP/jboss/icpConfig/icp.keystore;trustStorePassword=ingenix1;enabledProtocols=TLSv1,TLSv1.1,TLSv1.2</acceptor>
</acceptors>
```

Locate and uncomment the appropriate `<cluster-connections>` element (main or standby) by removing the surrounding comment characters that are highlighted below. There are two cluster-connections sections: one for the main node and one for the standby node. Be sure to uncomment the correct one. The example below is for the “main” node.

```xml
<!-- cluster connections for ha for Main node ... -->
<cluster-connections>
    <cluster-connection name="my-cluster">
        <address>jms</address>
    </cluster-connection>
</cluster-connections>
```

© 2020 Optum, Inc. 47
Search for occurrences of the `<address-full-policy>` XML element. Change the value from BLOCK to PAGE. There are four occurrences of this element, so be sure to change all four.

Save the file and exit the editor.

### 4.1.3 Configuring the jndi.properties File

| Note | No changes to the jndi.properties file are required for non-HA configurations (i.e., no standby Claim Connector). |

Located in the `<InstallDir>/conf` directory, the `jndi.properties` file has been pre-configured with four different configurations options:

- **Standalone (single node) without HTTPS/SSL** – This is the default “out-of-the-box” configuration and requires no changes.
- **High Availability without HTTPS/SSL enabled** – Only configuration for JMS server failover is required.
- **Standalone (single node) with HTTPS/SSL enabled** – Configuration changes for security plus application of SSL certificates is required.
- **High Availability with HTTPS/SSL enabled** – Configuration changes for security, application of SSL certificates, and JMS server failover must be configured.

After determining which of the above options you require, make a backup copy of `jndi.properties` and open it with a text editor.

1. Locate the `connectionFactory` line and uncomment it by removing the “#” from the front of the line.

Update this line by replacing the `<mainNameOrIP>` and `<standbyNameOrIP>` tags with the actual machine name or IP Address of the main and standby JMS server nodes.
Comment out all other connectionFactory lines by inserting a ‘#’ character at the beginning of the line.

**Original configuration** (HA without SSL)

```xml
### HA non-ssl settings ###
connectionFactory.ConnectionFactory=(
    tcp://<mainNameOrIP>:61616,
    tcp://<standbyNameOrIP>:61616
)
?consumerWindowSize=10240
&minLargeMessageSize=409600
&blockOnDurableSend=false
&preAcknowledge=true
&threadPoolMaxSize=-1
&ha=true
&retryInterval=1000
&retryIntervalMultiplier=1.0
&reconnectAttempts=-1
```

**New configuration**

```xml
### HA non-ssl settings ###
connectionFactory.ConnectionFactory=(
    tcp://<MainMachineName>:61616,
    tcp://<standbyMachineName>:61616
)
?consumerWindowSize=10240
&minLargeMessageSize=409600
&blockOnDurableSend=false
&preAcknowledge=true
&threadPoolMaxSize=-1
&ha=true
&retryInterval=1000
&retryIntervalMultiplier=1.0
&reconnectAttempts=-1
```

Save the file and exit the editor.

### 4.2 Configuration that Applies to All Nodes

The following subsections contain detailed instructions for configuration file modifications that must be performed on all nodes that participate in the claim processing system in order for them to participate in a multi-node High Availability (JMS clustered) system.

#### 4.2.1 Configuring the standalone.xml File

The standalone.xml file is located in both the `<InstallDir>`\jboss\icpConfig\standalone directory and `<InstallDir>`\jboss\standalone\configuration directory; however, the former is considered the “master” copy and the latter the “runtime” copy.

**Note**  The “runtime” version of the standalone.xml file may be overwritten by JBoss and all comments removed from the file; therefore, it is important to always follow the guidelines to preserve your original and edited versions of the file. This behavior is fully described in section 2: JBoss Configuration Procedures, which you should review before proceeding.

Configuration settings for Secure HTTP are normally commented out in the standalone.xml file.
1. Using a text editor (such as Notepad), open the standalone.xml file located in the 
<InstallDir>\jbos\icpConfig\standalone directory.

Search for the <system-properties> element (near the top of the file), then locate the
!-- JMS Settings --> heading. Make the following changes to properties in this section:

   a. Locate the JMSmain property and change its value to the name or IP Address of the
      main JMS server node.

   b. Locate the JMSstandby property and change its value to the name or IP Address of the
      standby JMS server node. Note that if multiple standby nodes are to be configured, you
      must add a line for each node where a standby JMS server will be running. Change the
      property names to “JMSStandby1”, “JMSStandby2”, etc.

   c. Locate the JMSha property and change its value from “false” to “true.” This enables High
      Availability (cluster communication) for the JMS server.

Original configuration

<table>
<thead>
<tr>
<th>Original configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!-- JMS Settings --&gt;</td>
</tr>
<tr>
<td>&lt;property name=&quot;JMSmain&quot; value=&quot;${localHostName}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;property name=&quot;JMSstandby&quot; value=&quot;&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;property name=&quot;JMSha&quot; value=&quot;false&quot;/&gt;</td>
</tr>
</tbody>
</table>

New configuration

<table>
<thead>
<tr>
<th>New configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!-- JMS Settings --&gt;</td>
</tr>
<tr>
<td>&lt;property name=&quot;JMSmain&quot; value=&quot;mainNodeHostname&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;property name=&quot;JMSstandby&quot; value=&quot;standbyNodeHostName&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;property name=&quot;JMSha&quot; value=&quot;true&quot;/&gt;</td>
</tr>
</tbody>
</table>

Locate the domain:messaging sub-system, then find the commented out <!--connector element
that is directly under it. Uncomment the connector section by removing the ‘!--’ and ‘--’
comment characters as shown below.

Original configuration

<table>
<thead>
<tr>
<th>Original configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!--connector</td>
</tr>
<tr>
<td>name=&quot;artemisStandby&quot;</td>
</tr>
<tr>
<td>factory-class=&quot;org.apache.activemq.artemis.core.remoting.impl.netty.NettyConnectorFactory&quot;&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;host&quot; value=&quot;${JMSstandby}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;port&quot; value=&quot;${JMSport}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;sslEnabled&quot; value=&quot;${JMSsslEnabled}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;keyStorePath&quot; value=&quot;${icpConfigDirPath}\${keystoreFileName}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;keyStorePassword&quot; value=&quot;${keystorePassword}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;trustStorePath&quot; value=&quot;${icpConfigDirPath}\${truststoreFileName}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;trustStorePassword&quot; value=&quot;${truststorePassword}&quot;/&gt;</td>
</tr>
</tbody>
</table>
| </connector-->

New configuration

<table>
<thead>
<tr>
<th>New configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;connector</td>
</tr>
<tr>
<td>name=&quot;artemisStandby&quot;</td>
</tr>
<tr>
<td>factory-class=&quot;org.apache.activemq.artemis.core.remoting.impl.netty.NettyConnectorFactory&quot;&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;host&quot; value=&quot;${JMSstandby}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;port&quot; value=&quot;${JMSport}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;sslEnabled&quot; value=&quot;${JMSsslEnabled}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;keyStorePath&quot; value=&quot;${icpConfigDirPath}\${keystoreFileName}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;keyStorePassword&quot; value=&quot;${keystorePassword}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;trustStorePath&quot; value=&quot;${icpConfigDirPath}\${truststoreFileName}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param name=&quot;trustStorePassword&quot; value=&quot;${truststorePassword}&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/connector&gt;</td>
</tr>
</tbody>
</table>
If multiple standby nodes are to be configured, you must add a `<connector>` element (total of 11 lines) for each node where a standby JMS server will be running. Change the “name” to “artemisStandby1”, “artemisStandby2”, etc. Note that the value for the “host” parameter will be picked up from the changes that were made to the `<system-properties>` element that were made in step 2.b above.

Locate the `<pooled-connection-factory` element that immediately follows the connector configuration. Modify the connectors attribute line so that the value is a space-separated list of the names of all connectors that have been defined in step 0 above. The example below is for two JMS servers. For a 3-node system, the value should be something like “artemisMain artemisStandby1 artemisStandby2”.

**Original configuration**

```
<pooled-connection-factory
  name="activemq-ra"
  entries="ConnectionFactory java:/JmsXA java:jboss/DefaultJMSConnectionFactory"
  connectors="artemisMain"
  min-large-message-size="10485760"
  block-on-durable-send="false"
  pre-acknowledge="true"
  ha="${JMSha}"  
  retry-interval="1000"
  retry-interval-multiplier="1.0"
  transaction="xa"/>
```

**New configuration**

```
<pooled-connection-factory
  name="activemq-ra"
  entries="ConnectionFactory java:/JmsXA java:jboss/DefaultJMSConnectionFactory"
  connectors="artemisMain artemisStandby"
  min-large-message-size="10485760"
  block-on-durable-send="false"
  pre-acknowledge="true"
  ha="${JMSha}"  
  retry-interval="1000"
  retry-interval-multiplier="1.0"
  transaction="xa"/>
```

Save the `standalone.xml` file and exit the editor.

Copy the `standalone.xml` file from the `<InstallDir>\jboss\icpConfig\standalone` directory to the `<InstallDir>\jboss\standalone\configuration` directory. This is the “runtime” location of the file and the file which is actually used by JBoss.

### 4.3 Configuration for “JBoss Only” Nodes

When many claim processing nodes are used in a system, it is most likely that only two of the nodes will use the Claim Connector with a JMS server and the other nodes will only need to run the core application which runs in the JBoss Application Server.

This section describes how to modify the configuration so that when the standard ICP Service desktop icon is used to start the system, the ICP Service will only start the ICP Engine service (JBoss) and will not start the ICP Connector service. The database management feature of the ICP Service is also disabled.
4.3.1 Configuring the services.properties File:
The services.properties file is located in the "<InstallDir>/conf" directory. Edit this file with a text editor to make the changes shown below.

1. Locate the product.services=import,coderepository,connector,jboss line near the top of the file. Comment it out by adding a ‘#’ character to the front of the line.

Locate the line underneath the "--- Jboss Only" heading and remove the ‘#’ character from the front of the line to uncomment it. Your lines should look like the following when done.

Original configuration

```
# All Services - Default #
#product.services=import,coderepository,connector,jboss
# Jboss Only     #
#product.services=jboss
# Connector Only #
#product.services=connector
```

New configuration

```
# All Services - Default #
product.services=jboss
# Jboss Only     #
#product.services=jboss
# Connector Only #
#product.services=connector
```

Save the file and exit the editor.

4.4 Non-HA Configuration

CM/CES 5.4 SP1 environments where High Availability (HA) is not enabled are referred to as “non-HA.” Non-HA means the environment does not have a standby JMS server (or connector) to fail over to if the main JMS server fails.

The instructions below describe how to configure a non-HA, multi-node CM/CES 5.4 SP1 environment.

4.4.1 Configuring the standalone.xml File

The instructions below apply to all nodes in the CM/CES environment.

The standalone.xml file is located in both the "<InstallDir>\jboss\icpConfig\standalone" directory and "<InstallDir>\jboss\standalone\configuration" directory; however, the former is considered the “master” copy and the latter the “runtime” copy.

1. Using a text editor (such as Notepad), open the standalone.xml file located in the "<InstallDir>\jboss\icpConfig\standalone" directory.

a. Search for the <system-properties> element (near the top of the file), then locate the <!-- JMS Settings --> heading.

b. Locate the JMSmain property and change its value to the name or IP Address of the main JMS server node.

Original configuration

```
<!-- JMS Settings -->
<property name="JMSmain" value="${localHostName}"/>
```
New configuration

```xml
<!-- JMS Settings -->
<property name="JMSmain" value="<jainNodeHostname"/>
```

2. Save the standalone.xml file and exit the editor.

3. Copy the standalone.xml file from the `<InstallDir>/jboss/icpConfig/standalone` directory to the `<InstallDir>/jboss/standalone/configuration` directory. This is the "runtime" location of the file and the file which is actually used by JBoss.

Repeat steps 1-3 for each node in the CM/CES environment.

### 4.4.2 Configuring the services.properties File:

**Note**
The instructions below apply to “JBoss Only” nodes. Do not update the `services.properties` file on the JMS server (connector) node.

The `services.properties` file is located in the `<InstallDir>/conf` directory. Edit this file with a text editor to make the changes shown below.

1. Locate the `product.services=import,coderepository,connector,jboss` line near the top of the file. Comment it out by adding a `#` character to the front of the line.

Locate the line underneath the `#--- Jboss Only` heading and remove the ‘#’ character from the front of the line. Your lines should look like the following when done.

**Original configuration**

```properties
# All Services - Default#
product.services=import,coderepository,connector,jboss
# Jboss Only   
#product.services=jboss
# Connector Only#
#product.services=connector
```

**New configuration**

```properties
# All Services - Default#
#product.services=import,coderepository,connector,jboss
# Jboss Only   
product.services=jboss
# Connector Only#
#product.services=connector
```

Save the file and exit the editor.

Repeat for each JBoss-only node in the CM/CES environment.

### 4.5 Considerations for IPv6

**Background:** Two internet addressing schemes are IP versions 4 and 6. IP version 4 uses the familiar internet addressing scheme a.b.c.d (for example, 192.168.100.1). However, the newer, more scalable version 6 uses 6 address bytes for a vastly larger number of potential unique addresses. This uses a less familiar format of a::b::c::d:e::f. IPv4 and IPv6 are both in use today, but many routers, switches, and computers are not very good at handling both at the same time.

**Recommendations:** When IPv6 is enabled for network interfaces, various problems can occur, especially with UDP multicasting. (Because UDP is no longer used when clustering JMS...
servers, this may no longer be a problem.) The preferred solution is to disable IPv6 and rely on IPv4 alone. Although this can be specified on the Networking tab of the properties dialog for each network connection, Windows does not fully honor this setting, and additional registry changes are required. Configuration parameters have been added to tell JBoss to prefer IPv4 over IPv6.

4.5.1 Disabling IPv6

Configuration Example: IPv6 can be disabled in the Windows registry via the following steps:

1. Disable the IPv6 protocol from all network adapters by unchecking that option in the network control panel.

2. Completely turn off all attempts by Windows to use IPv6 by setting the registry entry as follows. (This applies to all versions of Windows. Reboot after making the change.)

3. Note that the “DisabledComponents” entry may not already exist. If so, create it by right-clicking on the right-hand panel of the “regedit” screen (after navigating to the Parameters directory) and select New → Binary value from the shortcut menu. This value is a set of bit flags for the IPv6 components that should be disabled, so by setting the value to 0xFFFFFFFF, all components are disabled.

**Configuration File Location**

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\TCPIP6\Parameters\DisabledComponents
```

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\TCPIP6\Parameters\DisabledComponents
(REG_DWORD) = 0xFFFFFFFF
```

4.6 Web Session Load Balancing

**Background:** The ICP JBoss cluster is not configured for session replication by default. After a user logs into the user interface, all subsequent requests must be sent to the node where the session was created. The load balancer may send the initial request for login to any of the nodes in the cluster, but it must stick to the same node for all other requests associated with that login. Otherwise, errors will occur (visible in the jboss.server.log) when the request is sent to a node that knows nothing about the web session.

**Recommendations:** Load balancers will have a “sticky session” configuration option. It must be used in relation to the web network traffic to the JBoss cluster nodes. (Specifics are outside of the scope here.)

**Additional Information:** The following information may help when considering load balancing.

4.6.1 STATIC Load Balancing Methods

**Round Robin** – Evenly distributes requests to all available pool members.

**Ratio** – Ratio allows each server to be assigned a ratio value. This is useful for pool members that have greater or lower computing resources then others. Example: Ratio 3:2:1:1. Based upon 8 requests, 3 requests would go to 1, 2 to 2 then 1 to 1.

4.6.2 DYNAMIC Load Balancing Methods

**Least Connections** – Traffic is balanced to servers with the least total of current connections. Fastest connections are distributed to pool members based upon server response time.
**Observed** – This method is the same as ratio, but the ratio is assigned by BIG-IP. Each ratio is calculated based on the total number of connections currently active on each pool member. A pool member with a lower-than-average connection count is assigned a ratio of 3. A pool member with a higher-than-average count is given ratio of 2.

**Predictive** – Predictive is similar to observed but ratios are assigned using much more aggressive ratio values. A pool member with a lower than average connection count is assigned a ratio of 4. A pool member with a higher-than-average count is given ratio of 1.

### 4.6.3 Sticky Session/Session Affinity

| Note | This information is specific to the “BIG-IP” load balancer. Other vendors will have a similar option. |

When you configure the BIG-IP system to manage HTTP traffic, you can also implement simple session persistence, also known as **source address affinity persistence**. Source address affinity persistence directs session requests to the same server based solely on the source IP address of a packet. To implement source address affinity persistence, the BIG-IP system offers a default persistence profile that you can implement. Just as for HTTP, you can use the default profile or you can create a custom simple persistence profile.

### 4.7 Operational Considerations

Due to the software component configuration changes that are introduced in 5.4 Service Pack 1, several changes have occurred that may impact the operation of the overall system. These are discussed in the following sections.

#### 4.7.1 Location of JMS Journal Files

The JMS Server (Artemis) has been integrated with the JBoss Application Server up until this point. As such, the data files where JMS messages are persisted (journal files) were located in the JBoss folder but are now located in a new folder for Artemis.

**Original location**

\<InstallDir\>\jboss\standalone\data\journal

**New location**

\<InstallDir\>\artemis\data\journal

This directory path is commonly used in Windows batch scripts that are used to start up the ICP services that archive log files and delete temporary data including the JMS journal files. For these scripts, it is fine to delete the entire data directory (not just the journal directory).

#### 4.7.2 Operating an HA System

Operationally, the new HA configuration is very similar to the operation of previous clustered configurations, but with a number of improvements. Here are some things you should be aware of:

1. The **ICP Connector** service now starts second after the **ICP Service** whereas previously it was the last service to start. This is because JBoss requires a JMS Server to be running when it starts up, so **ICP Engine** (JBoss) is now the last service to start.
2 The item above implies that before “JBoss only” nodes (nodes where the Claim Connector and JMS Server are not running) are started, at least one node with a JMS Server must be started.

3 The “JBoss only” nodes are all configured exactly the same. They all point to the same list of JMS servers.

4 When one node in the cluster fails, it does not affect the operation of any other node. (This is not true of the older system that used JBoss clustering.)

5 It is possible to add a new node into a running multi-node cluster, but it does cause message delivery to all nodes to stop for between 20 to 30 seconds while the new node is coming online, after which normal claim processing resumes. This delay gets added to the total claim processing time from the viewpoint of the sending system and can cause claim timeouts. Because of this, it is recommended that new nodes be added during a maintenance window when the entire cluster can be stopped and restarted.

6 Another issue related to adding a new node to a running system relates to the prevention of Parallel Patient claim Processing (PPP). (See section Prevention of Parallel Patient Claim Processing for details.) Because all claims for the same Patient ID are assigned to the same node, any claim for a patient that was previously “seen” and allocated to any of the running nodes will never be sent to the new node after it is added to the cluster. This will result in an uneven workload distribution. (The new node will not get its “fair share” of claims.) The “memory” for patient claim distribution is reset by stopping the JMS server and restarting it; therefore, it is best to stop the entire cluster and restart it along with the new node during a maintenance window.
5 Setting up Secure Communications

5.1 Introduction to Secure Communication

5.1.1 Secure Communication Protocols

Modern secure network communication relies on strong cryptography provided by special communication protocols such as TLS (Transport Layer Security) which is the latest version of SSL (Secure Socket Layer). Prior to sending any important data, these protocols engage in an exchange that allows them to agree upon a cipher suite which is a set of algorithms that have all undergone extensive cryptanalysis for security and strength. A cipher suite will include an asymmetric (or “public-key”) cryptography algorithm, a symmetric cryptography algorithm and a hashing algorithm. All of these are essential to establishing a highly secure communication channel that can withstand attempts to decrypt network traffic across an untrusted network.

In addition to establishing a secure channel of communication, the other essential function of the TLS/SSL encryption protocols is to provide a way for each machine to cryptographically verify the identity of the end point it is communicating with. This is accomplished by the use of digital certificates which are summarized in the next section.

In many instances, this verification is often only performed in a single direction where the client (the computer initiating the connection) verifies the identity of the server (the computer receiving the connection request), but the server does not verify the identity of the client. Most public facing internet sites behave like this. For example, when you connect to your bank using HTTPS (HTTP over SSL) and after negotiating a secure connection, your browser will verify the identity of the server by means of a digital certificate that is installed on the server. Upon completion of the verification, the browser will typically display a “lock” icon and you will then see the login page. In this instance, the server did not cryptographically verify your browser as a valid client; instead, it requires you to enter credentials (username and password) in order to verify your identity and allow you to log in and use its services.

However, for secure communication between business servers, verification by digital certificate is often performed in both directions. (The client verifies the digital certificate on the server and the server verifies the digital certificate on the client.) This scenario is called “mutual authentication”. This allows the client to securely connect to the server without a login screen or the input of credentials.

5.1.2 Digital Certificates and Keys

A digital certificate (public key certificate / identity certificate) is an electronic document (small data file) used to prove ownership of a private key. The certificate includes a public key, information about its owner's identity, and may contain the digital signature of a trusted entity called a Certificate Authority (CA) that has verified that the certificate's contents are correct. If the certificate's signature is valid (i.e. it has not been tampered with), and the system examining the certificate has been configured to trust the signer (the CA), the system will then use the public key in the certificate to encrypt a challenge to the remote system that presented the key. When the remote system responds correctly to the challenge, the system issuing the challenge knows that the remote system possesses the corresponding private key and is therefore cryptographically verified to be trustworthy.

The interrelationship of certificates and trusted certificate authorities that sign those certificates form a Public-Key Infrastructure (PKI) scheme which allows for a “web of trust” to be created on
the modern Internet or within a corporate intranet. It creates an environment where communicating parties trust one another.

Note that digital certificates are verified using a “chain of trust”. The trust anchor for the digital certificate is the Root Certificate Authority (CA).

A certificate authority can issue multiple certificates in the form of a tree structure. A root certificate is the topmost certificate of the tree, the private key of which is used to sign other certificates.

There are many elements to the public key infrastructure that makes up the modern internet that are beyond the scope and requirements to configure and use CM/CES. This document will constrain its discussion to the use cases encountered when installing and running the CM/CES software. This includes:

1. Using self-signed keys to get secure communications up and running quickly
2. Importing keys signed generated and signed by a certificate authority that is typically under the control of the company operating the CM/CES instance

5.1.3 Keystore and Truststore

Both keystore and truststore refer to files where digital certificates and keys are stored. A keystore contains private keys and the certificates with their corresponding public keys. A truststore contains certificates from other parties that you expect to communicate with, or from Certificate Authorities that you trust to identify other parties. It is possible for the keystore and truststore to be the same file. In an SSL handshake, the truststore is used to verify credentials and the purpose of keystore is to provide credentials.

There are two sets of keystore and truststore files that are used by CM/CES software:

1. **Secure communication for the Claim Connector**. This includes communication between a source of claims and the Claim Connector as well as communication between the Claim Connector and the JMS Server. Of these two, only the Claim Connector and a claim source. CM/CES software typically receives claims for analysis from a client’s Practice Management System (providers) or an Adjudication System (payers). In other cases, batches of claims are transferred as text files and placed in a directory where the XMP Batchloader can process them and send the claims to the Claim Connector. In all of these cases, the network connection between the claim sender and the Claim Connector should be secured with SSL digital certificates.

2. **Secure communication for the JBoss Application Server**. This includes communication between an end-user’s web browser and the web server that runs in JBoss as well as JMS communication between the JMS Server and JBoss where the claim analysis is performed. It also includes any outbound connections that are made by the JBoss Application Server to any external systems.
The following diagram identifies which software components use which set of keystore and truststore files when applying SSL certificates.

Figure 4 – Logical System Configuration for High Availability

5.1.4 The Java Keytool Command

Keytool is a key and certificate management tool that is used to manipulate Java Keystores and is included with Java. A Java Keystore is a container for digital certificates and key pairs necessary for negotiating secure TLS/SSL connections. Its entries are stored encrypted and protected by a keystore password. A keystore entry is identified by an alias.

5.1.5 Keytool Summary

- Allows the import of public keys and certificates such that a system will begin to trust them for communication.
- Enables creation of self-signed keys that can be used for authentication with other nodes.
- Imports a key pair issued by another system that may have been signed by a corporate or public CA such that the system can become a member of a corporate “web of trust”.

5.2 Getting Started

The instructions in the following sections are intended for administrators with a good understanding of how to install and configure a CM/CES product. The instructions cover both standalone and clustered environments, but they are primarily for standalone (non-clustered) installations. For those setting up CM/CES software in a clustered configuration, there are annotated subsections with special procedures for setting up HTTPS on a clustered system. **Make sure these special instructions are followed only if you are configuring a clustered system.** These instructions are the same for both newly installed systems and upgraded systems. If you have just upgraded from an older version of CM/CES and previously had HTTPS configured, most of this will still apply with the exception of the creation of your keys, certificates, and keystores. You should be able to copy the icp.keystore file from your backup location rather than creating a new one.
5.2.1 Upgrade Considerations

| Note | When 5.4 Service Pack 1 is applied to an existing 5.4 installation of CM/CES, a new release of the JBoss Application Server (Wildfly-10) is installed and the icp.keystore file is saved in the BackupFiles\jboss\icpConfig directory where the service pack was installed. This file must be copied to the <InstallDir>\jboss\icpConfig directory for the new instance of JBoss to restore the SSL keys and certificates that were previously installed. HTTPS connections should be functional. |

If you are upgrading the CM/CES product that has already been configured to use HTTPS, you should already have set up the keystore and will also already have appropriate digital certificates from a signing authority in your icp.keystore file. If you are sure this applies to your system, you can skip forward to the section Importing a Signed Certificate (.pfx file)

In the Command Prompt window, type the command below to apply the certificates. For each command, a keystore password must be supplied along with confirmation that the certificates are trusted. There may be a requirement to import more than the root certificate and your own certificate (from your certificate authority). This example assumes a .pfx certificate file is being used. (It may also have a different file extension such as .p12.)

- Use the keytool -list command to view the content of the certificate. The alias name of the root authority certificate should be included in the output.

```
keytool -printcert -v -file <certificate>
EXAMPLE keytool -printcert -v -file icpCMprod01.pfx
```

- Create a Java keystore from the .pfx file. This command assumes the .pfx file password is Optum2018 and the new keystore file password will be T4dpo!e79. Adjust as necessary. In all of these examples, the password flags and values can be left off of the command, and the user will be prompted to enter them. This example is essentially copying a certificate that is identified by its alias from one keystore to another.

```
keytool -importkeystore -srckeystore <certificate_file> -srcstoretype pkcs12 -srcstorepass <certfile_pwd> -destkeystore <dest_keystore> -deststoretype JKS -deststorepass <keystore_pass> -srckeypass <source_key_pass> -dstkeypass <dest_key_pass>
EXAMPLE keytool -importkeystore -srckeystore icpCMprod01.pfx -srcstoretype pkcs12 -srcstorepass MyVeryLongPassword#246 -destkeystore icp.keystore -deststoretype JKS -deststorepass T4dpo!e79 -srckeypass "icp CMprod01" -srckeypass MyVeryLongKeypassword#17 -dstkeypass T4dpo!e79
```

Note that the above example shows how values (source alias) must be inside double quotes when they contain spaces. Also, in this case the source keystore and key passwords are different whereas the destination keystore and destination key passwords are made to be the same.

- Use the keytool –list command shown in the example below to view the contents of your icp.keystore file to verify contents.

```
Keytool -v -list -keystore icp.keystore -storepass T4dpo!e79
```

5.2.1.1 Importing Certificates from a Java Keystore (.jks file)

In the Command Prompt window, type the command below to apply the certificates. For some commands, you are asked to confirm that you trust the certificates. There may be a requirement to import more than the root certificate and your own certificate (from your certificate authority). This example assumes a .jks Java keystore file is being used.
Use the `keytool -list` command to view the content of the keystore that contains the signed certificate. The alias name of the root authority certificate should be included in the output.

```
keytool -list -v -keystore <source_keystore>
```

**EXAMPLE**
```
keytool -list -v -keystore icpCMprod01.jks
```

Import a certificate from the `.jks` file into the ICP keystore file. This command assumes the `.jks` file password is `Optum2018` and the new keystore file password will be `T4dpo!e79`. Adjust as necessary. In all of these examples, the password flags and values can be left off of the command, and the user will be prompted to enter them.

```
keytool -importkeystore -srckeystore <certificate_file> -srcstoretype JKS -srcstorepass <certfile_pwd> -destkeystore <dest_keystore> -deststoretype JKS -deststorepass <keystore_pass> -srcalias <cert_alias>
```

**EXAMPLE**
```
keytool -importkeystore -srckeystore icpCMprod01.jks -srcstoretype JKS -srcstorepass Optum2018 -destkeystore icp.keystore -deststoretype JKS -deststorepass T4dpo!e79 -srcalias icpCMprod01
```

Import a private key from the `.jks` file into the ICP keystore file. Note that in this case, source and destination key passwords must also be specified in addition to the source to the destination keystore passwords.

```
keytool -importkeystore -srckeystore <certificate_file> -srcstoretype JKS -srcstorepass <certfile_pwd> -destkeystore <dest_keystore> -deststoretype JKS -deststorepass <keystore_pass> -srcalias <cert_alias> -srckeypass <source_key_pass> -dstkeypass <dest_key_pass>
```

**EXAMPLE**
```
keytool -importkeystore -srckeystore icpCMprod01.pfx -srcstoretype pkcs12 -srcstorepass MyVeryLongPassword#246 -destkeystore icp.keystore -deststoretype JKS -deststorepass T4dpo!e79 -srcalias KeyCMprod01 -srckeypass MyVeryLongKeyPassword#17 -dstkeypass T4dpo!e79
```

Use the `keytool –list` command shown in the example below to view the contents of your icp.keystore file to verify contents.

```
Keytool –v –list –keystore icp.keystore -storepass T4dpo!e79
```

### 5.2.2 Distributing Signed Certificates

Signed certificates commonly need to be distributed to another Java truststore (i.e., adding to the truststore for the Claim Connector, exchanging between connector and the customer software that sends claims to CM/CES, or adding it to a browser’s trusted certificates). The root authority certificate may also need to be distributed.

#### 5.2.2.1 Importing Certificates into a Truststore

The following commands can be used to import a certificate into a truststore. Note that it is essentially the same thing as importing into the icp.keystore file.

- Use the `keytool -importcert` command to import the signed certificate.

```
keytool -importcert -file <certificate_file> -alias <cert_alias> -keystore <truststore> -storepass <truststore_pass>
```

**EXAMPLE**
```
keytool -importcert -file icpCMprod01.crt -alias icpCMprod -keystore truststore.ks -storepass T4dpo!e79
```

- Use the `keytool -importcert` command to import the root authority certificate.

```
keytool -importcert -file <CAroot_certificate> -alias <CAroot_alias> -keystore <truststore> -storepass <truststore_pass>
```

© 2020 Optum, Inc.
Use the `keytool -list` command to view the contents of your `truststore.ks` file to verify contents.

```bash
test -v -list -keystore truststore.ks -storepass T4dpo!e79
```

The intermediate/signing certificate does not need to be imported into the truststore along with the signed certificate. If certificate chain errors are experienced after importing the Root and Signed certificate, the signed certificate can be imported by following the same import command, but changing the file and alias values. The alias is a description that is client configurable to identify the certificate, like `icpCMprod01`.

### 5.2.3 Using a Workspace Directory

Create a directory where the following tasks can be performed (you can use a Windows Explorer window to create it). **Do not create it inside the CM/CES installation path. This will prevent anything from accidentally being deleted during a subsequent upgrade or uninstall.** Once the tasks are complete, follow the instructions on where to place the new or modified files. This directory will be referred to as the **workspace directory**.

| Note | Here `<InstallDir>` refers to the root directory of the CM/CES software installation (e.g., `C:\Optum\ICP`), and the workspace directory is `C:\icpworkspace`. |

1. On the machine where CM/CES software is installed, open a **Command Prompt** window.
2. Navigate to the workspace directory. (e.g., `cd c:\icpworkspace`)
3. The `Path` environment variable must be modified for this **Command Prompt** window to ensure that Java commands included with the JRE (Java Runtime Environment) that was installed with CM/CES software can be found. This is important since it includes the `keytool` command.
   a. In the **Command Prompt** window opened in the previous step, enter the following command: `Set PATH=<InstallDir>\jre\bin;%PATH%`

| Note | It is assumed that you are setting up HTTPS for the first time on this system. Otherwise, locate your existing `icp.keystore` file and skip to section 5.4 Updating the Keystore for Cluster Nodes. |

If you have any existing keys/certificates in your `icp.keystore` file and want to add a new key to this same file, copy your existing file to the **workspace directory**. If you are starting from scratch to create new keys, this process will create the `icp.keystore` file for you.

### 5.2.4 Summary of Parameters for the Keytool Utility

The `keytool` command is a utility program that is delivered with the **Java Development Toolkit (JDK)**. It is the primary tool that is used for managing digital certificates, keystores, and truststores for Java software. The following is a brief description of the parameters that can be used with the `keytool` command.
<table>
<thead>
<tr>
<th>Keytool Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ext san=dns:&lt;hostname&gt;</td>
<td><code>&lt;hostname&gt;</code> is the DNS host name of this machine (i.e., The name that is used in a browser to reach the machine). This parameter may be repeated multiple times with different hostname values if the machine is known by different aliases (i.e., including “localhost”).</td>
</tr>
<tr>
<td>-ext san=ip:&lt;ipaddress&gt;</td>
<td><code>&lt;ipaddress&gt;</code> is the IP Address of this machine (i.e., 10.5.0.201). This parameter should only be included if your IP Address is static. If the IP may change (i.e., if it assigned by DHCP), it should not be included. If this machine has multiple network interfaces, this parameter may be repeated multiple times for the IP Address of each network interface.</td>
</tr>
<tr>
<td>-alias &lt;keyalias&gt;</td>
<td><code>&lt;keyalias&gt;</code> must be a unique name. It is recommended that you use <code>icp&lt;hostname&gt;</code> (replacing <code>&lt;hostname&gt;</code> with your actual hostname) since it uniquely identifies the key and associates it with both the machine and the application.</td>
</tr>
<tr>
<td>-validity &lt;days&gt;</td>
<td><code>&lt;days&gt;</code> is the number of days the certificate is valid. (e.g., use 365 for one year.)</td>
</tr>
</tbody>
</table>
| -dname | A list of “Distinguished Name” name=value pairs enclosed in double-quotes. The following names may be used:  
  - **cn** Common Name. `<hostname>` Use the fully qualified `HostName/Path` used in DNS lookups that the machine will be running on (e.g., `HostName.Example.com`).  
  - **o** Organization. `<organization>` The name you specify for the Organization field should be the legal name for your organization that is registered with the appropriate city, state, or country/region authority. The legal name of the organization must be used in the Organization field.  
  - **ou** Organizational Unit. `<organizationUnit>` The Organizational Unit field can be used to differentiate between different divisions within an organization, for example “Internet Security Unit” or “Human Resources.” This field is also recommended to be used for specifying a DBA (Doing Business As...) value.  
  - **l** Location. `<location>` Denotes the city that the organization resides in.  
  - **st** State or Province. `<state>` Specifies where the organization is physically located. If your organization is incorporated in Delaware but has a DBA (Doing Business As...) within California, use California. The State or Province field should not be an abbreviated field. For example, “CA” is not a valid state name. “California” is the proper state name.  
  - **c** Country/Region. `<country>` The X.500 Naming Scheme standard requires a 2-character country/region code. The country/region code for the United States is US; the country/region code for Canada is CA. |
5.3 Managing Digital Certificates

**Cluster Note**  For a clustered system, a Public Key Certificate for each node in the cluster (the master node and each slave node) will need to be acquired. Follow the steps below for each node, taking special care to use the correct hostname for the node and to create unique keystore aliases for each node.

### 5.3.1 Obtain a Private/Public Key Pair

To use secure connections a server must have a private key/public key pair. There are three ways to achieve this which are documented in the following subsections.

1. Request a certificate signing from a Certificate Authority. ([Section 5.3.1.1](#))
2. Import a key pair generated and signed by an external system. ([Section 5.3.1.2](#))
3. Use a self-signed certificate ([Section 5.3.1.3](#))

#### 5.3.1.1 Request a Certificate from a Certificate Authority

There are many reputable third party certificate authorities that can provide signed certificates for production use (IdenTrust, VeriSign, Comodo, DigiCert, GoDaddy, Symantec, etc.). A fee is typically charged for each certificate. A certificate is obtained by creating a Certificate Signing Request using the `keytool` utility, then sending it to the selected certificate authority for signing.

**Create a Certificate Keystore and Public/Private Key Pair**

Open a *Command Prompt* window in the workspace directory and update the PATH environment variable as previously described (`set PATH=<InstallDir>\jre\bin;%PATH%`).

Enter the following `keytool` command, modified with your specific configuration values as explained in section 5.2.4 above. This command will add your private/public key pair to your existing `icp.keystore` file if it exists, or it will create a new file with the key pair.

```bash
keytool -genkeypair -ext san=dns:<hostname> -ext san=ip:<ipaddress> -alias <keyalias> -keyalg RSA -keysize 2048 -keystore icp.keystore -storepass <mypasswd> -keypass <mypasswd> -validity <days> -dname "cn=<hostname>, o=<organization>, ou=<organizationUnit>, l=<location>, st=<state>, c=<country>"
```

**Note**  The password `<mypasswd>` specified in this command example must match the password that is specified in various other configuration files (`icp.wrapper.conf`, `connector.wrapper.conf`, `standalone.xml`). Check your configuration files and ensure that they match.

**Example**  

```bash
keytool -genkeypair -ext san=dns:CMprod01 -alias icpCMprod01 -keyalg RSA -keysize 2048 -keystore icp.keystore -storepass ingenix1 -keypass ingenix1 -validity 365 -dname "cn=CMprod01, o=optum, l=westvalley, st=utah, c=us"
```

The `-list` option to the `keytool` command can be utilized to view the contents of this keystore.

```bash
keytool -v -list -keystore icp.keystore -storepass ingenix1
```

The `-list` option to the `keytool` command can be utilized to view the contents of this keystore.
Generate a Certificate Signing Request

Next, enter the following command, modified with your specific configuration values.

```
keytool -certreq -keyalg RSA -alias <yourKeyAlias> -file <yourKeyAlias>.csr -keystore icp.keystore -storepass ingenix1
```

**EXAMPLE**
```
keytool -certreq -keyalg RSA -alias icpCMprod01 -file icpCMprod01.csr -keystore icp.keystore -storepass ingenix1
```

A file named `<yourKeyAlias>.csr` is then generated and `<yourKeyAlias>` is replaced by your alias name that was used when running the command. This is your Certificate Signing Request file that must be submitted to an authorized certificate authority to obtain your public certificate.

Submit the Certificate Signing Request

The `.csr` file generated above is a certificate request file. Submit this file to a certificate authority. Details on how to submit the CSR are available from the specific certificate authority of choice.

5.3.1.2 Use a Corporate Certificate Authority to Obtain a Certificate

It is common for large enterprises to act as their own certificate authority. They will typically use software such as Venafi, which is an enterprise key and certificate management system. In other cases, a product such as the Microsoft Certificate Authority is used.

When requesting a keystore from a corporate CA, the following options are recommended (some of which may not be available).

1. Request that it be in Java Keystore (JKS) format. Many other formats are possible (PEM, DER, P7B, P7C/PKCS#7, PFX, P12/PKCS#12), but of these PFX, PKCS7, PKCS11, or PKCS12 are preferred because they are supported by the keytool utility.
2. Use **SHA-256** for the hash algorithm.

3. The keystore should also include the **Root Chain**. This is all certificates that form a chain from the certificate being issued up to the root authority certificate. This is needed in order to have your certificate verified as “valid” when it is not issued by a well-known CA.

4. The keystore should also include the **Private Key**.

5. Make sure that the same password is used for both the key and the keystore.

When the keystore or certificate file is obtained it will have a file extension that indicates the type of the file (ideally it is ‘.jks’). If it is in the form of a Java Keystore, copy this file to the workspace folder. It can be used there to transfer contents into an existing icp.keystore file, or it can be renamed and act as the initial icp.keystore file.

**5.3.1.3 Create a “Self-Signed” Certificate.**

This is done by creating a key pair using the keytool utility as described in the first step of section **5.3.1.1 - Request a Certificate from a Certificate Authority**. However, since the use of self-signed certificates is considered to be inappropriate for a production environment, the details for this option are not covered in this document.

**5.3.2 Import the Signed Certificate into the Keystore**

After you receive the signed certificate from your certificate authority, it must be imported into the icp.keystore file that contains the original private/public key pair located in the working directory (e.g., c:\icpworkspace\icp.keystore). If the certificate is in the form of a Java Keystore file (has a .jks file extension), it can be used directly by renaming it as the icp.keystore file. (This latter case only applies if this is the initial certificate for the keystore.)

To apply the certificate, follow the instructions provided in the response from the certificate authority. The instructions are usually organized by “Server Type,” which for ICP would be “Apache.”

There may be multiple certificates that must be imported in a specific order. For the CM/CES product, the keystore file and truststore file are combined into a single “java keystore” file named icp.keystore. (The exception to this is the claim interface. A truststore is still needed in that case because of its use of mutual authentication.) All required certificates (root certificate from the certificate authority, any required intermediate certificates, and your signed certificate) are all contained in the same file (icp.keystore). ICP does not use the java cacerts file.

For the examples that follow, begin by opening a Command Prompt window and navigate to the workspace directory (see section **5.2.3**). The required keytool commands are executed from within that directory. Also make sure that the PATH environment variable has been updated to include the directory where the keytool utility is located.

**EXAMPLE**

```
Set PATH=<InstallDir>\jre\bin;%PATH%
```

**5.3.2.1 Importing a Signed Certificate (.crt file)**

**Note** The following is an example only. Please follow the specific instructions from the signing authority.

In the Command Prompt window, type the command below to apply the certificates. For each command, a keystore password must be supplied along with confirmation that the certificates are trusted. There may be a requirement to import more than the root certificate and your own
certificate (from your certificate authority). This example assumes a .crt certificate file is being used. (It may also have a different file extension.)

- Use the **keytool -printcert** command to view the content of the certificate. The alias name of the root authority certificate should be included in the output.

```
keytool -printcert -v -file <certificate>
```

**EXAMPLE**
```
keytool -printcert -v -file icpCMProd01.crt
```

- Import the root certificate of the CA with what CSR has signed, replacing <CARootCert> with an alias name that correctly identifies the certificate for you (e.g., AcmeRootCert).

```
keytool -importcert -file rootcert.crt -alias <CArootCert> -keystore icp.keystore
```

- Import your signed certificate with the same alias as your original key. The alias name must match the alias of your key. If you do not remember, use the **keytool –list** command shown in the example below to view the contents of your icp.keystore file to verify the alias name for your private key.

**EXAMPLE**
```
:: View the contents of the icp.keystore file.
Keytool -v -list -keystore icp.keystore
:: Import your signed certificate.
keytool -import -alias <yourKeyAlias> -file <CACertificateName>.crt -keystore icp.keystore
```

### 5.3.2.2 Importing a Signed Certificate (.pfx file)

In the *Command Prompt* window, type the command below to apply the certificates. For each command, a keystore password must be supplied along with confirmation that the certificates are trusted. There may be a requirement to import more than the root certificate and your own certificate (from your certificate authority). This example assumes a .pfx certificate file is being used. (It may also have a different file extension such as .p12.)

- Use the **keytool -list** command to view the content of the certificate. The alias name of the root authority certificate should be included in the output.

```
keytool -printcert -v -file <certificate>
```

**EXAMPLE**
```
keytool -printcert -v -file icpCMProd01.pfx
```

- Create a Java keystore from the .pfx file. This command assumes the .pfx file password is Optum2018 and the new keystore file password will be T4dpo!e79. Adjust as necessary. In all of these examples, the password flags and values can be left off of the command, and the user will be prompted to enter them. This example is essentially copying a certificate that is identified by its alias from one keystore to another.

```
keytool -importkeystore -srckeystore <certificate_file> -srcstoretype pkcs12 -srcstorepass <certificate_pwd> -destkeystore <dest_keystore> -deststoretype JKS -deststorepass <keystore_pass> -srclalias <cert_alias> -srckeypass <source_key_pass> -dstkeypass <dest_key_pass>
```

**EXAMPLE**
```
keytool -importkeystore -srckeystore icpCMprod01.pfx -srcstoretype pkcs12 -srcstorepass MyVeryLongPassword#246 -destkeystore icp.keystore -deststoretype JKS -deststorepass T4dpo!e79 -srclalias "icp CMprod01" -srckeypass MyVeryLongKeyPassword#17 -dstkeypass T4dpo!e79
```

Note that the above example shows how values (source alias) must be inside double quotes when they contain spaces. Also, in this case the source keystore and key passwords are different whereas the destination keystore and destination key passwords are made to be the same.
• Use the `keytool -list` command shown in the example below to view the contents of your `icp.keystore` file to verify contents.

```
keytool -v -list -keystore icp.keystore -storepass T4dpo!e79
```

5.3.2.3 Importing Certificates from a Java Keystore (.jks file)

In the *Command Prompt* window, type the command below to apply the certificates. For some commands, you are asked to confirm that you trust the certificates. There may be a requirement to import more than the root certificate and your own certificate (from your certificate authority). This example assumes a `.jks` Java keystore file is being used.

• Use the `keytool -list` command to view the content of the keystore that contains the signed certificate. The alias name of the root authority certificate should be included in the output.

```
keytool -list -v -keystore <source_keystore>
```

**EXAMPLE**
```
keytool -list -v -keystore icpCMProd01.jks
```

• Import a certificate from the `.jks` file into the ICP keystore file. This command assumes the `.jks` file password is Optum2018 and the new keystore file password will be T4dpo!e79. Adjust as necessary. In all of these examples, the password flags and values can be left off of the command, and the user will be prompted to enter them.

```
keytool -importkeystore -srckeystore <certificate_file> -srcstoretype JKS -srcstorepass <certfile_pwd> -destkeystore <dest_keystore> -deststoretype JKS -deststorepass <keystore_pass> -srcalias <cert_alias>
```

**EXAMPLE**
```
keytool -importkeystore -srckeystore icpCMprod01.jks -srcstoretype JKS -srcstorepass Optum2018 -destkeystore icp.keystore -deststoretype JKS -deststorepass T4dpo!e79 -srcalias icpCMprod01
```

• Import a private key from the `.jks` file into the ICP keystore file. Note that in this case, source and destination key passwords must also be specified in addition to the source and destination keystore passwords.

```
keytool -importkeystore -srckeystore <certificate_file> -srcstoretype JKS -srcstorepass <certfile_pwd> -destkeystore <dest_keystore> -deststoretype JKS -deststorepass <keystore_pass> -srcalias <cert_alias> -srckeypass <source_key_pass> -dstkeypass <dest_key_pass>
```

**EXAMPLE**
```
keytool -importkeystore -srckeystore icpCMprod01.pfx -srcstoretype pkcs12 -srcstorepass MyVeryLongPassword#246 -destkeystore icp.keystore -deststoretype JKS -deststorepass T4dpo!e79 -srcalias KeyCMprod01 -srckeypass MyVeryLongKeyPassword#17 -dstkeypass T4dpo!e79
```

• Use the `keytool –list` command shown in the example below to view the contents of your `icp.keystore` file to verify contents.

```
Keytool -v -list -keystore icp.keystore -storepass T4dpo!e79
```

5.3.3 Distributing Signed Certificates

Signed certificates commonly need to be distributed to another Java truststore (i.e., adding to the truststore for the Claim Connector, exchanging between connector and the customer software that sends claims to CM/CES, or adding it to a browser’s trusted certificates). The root authority certificate may also need to be distributed.

5.3.3.1 Importing Certificates into a Truststore

The following commands can be used to import a certificate into a truststore. Note that it is essentially the same thing as importing into the `icp.keystore` file.
Use the `keytool -importcert` command to import the signed certificate.

```
keytool -importcert -file <certificate_file> -alias <cert_alias> -keystore <truststore> -storepass <truststore_pass>
```

**EXAMPLE**

```
keytool -importcert -file icpCMprod01.crt -alias icpCMprod -keystore truststore.ks -storepass T4dpo!e79
```

Use the `keytool -importcert` command to import the root authority certificate.

```
keytool -importcert -file <CAroot_certificate> -alias <CAroot_alias> -keystore <truststore> -storepass <truststore_pass>
```

**EXAMPLE**

```
keytool -importcert -file RootCA.crt -alias RootCA -keystore truststore.ks -storepass T4dpo!e79
```

Use the `keytool -list` command to view the contents of your `truststore.ks` file to verify contents.

```
keytool -v -list -keystore truststore.ks -storepass T4dpo!e79
```

The intermediate/signing certificate does not need to be imported into the truststore along with the signed certificate. If certificate chain errors are experienced after importing the Root and Signed certificate, the signed certificate can be imported by following the same import command, but changing the file and alias values. The alias is a description that is client configurable to identify the certificate, like `icpCMprod01`.

### 5.4 Updating the Keystore for Cluster Nodes

For a multi-node system, request and receive a signed certificate for each node that participates in the system. In addition to applying the signed certificate that was created for the local node, apply the public certificates from both nodes where the JMS server will be running (main and standby nodes) to all the other nodes, including to each other. A “JBoss only” node does not need the certificate for other “JBoss only” nodes; it only requires the public certificates of the two nodes where the JMS server is running. Follow these steps:

1. After importing the certificate for the local node into the keystore as described in the previous section, copy the certificate files for the JMS server nodes to the local **workspace directory**.

   Import the certificate for each JMS Server node into the local keystore using the following command:

```
keytool -importcert -file <otherhostcertificate.crt> -alias icp<otherhostname> -keystore icp.keystore
```

**Note** The previous command assumes the user:

1. Is using the Command Prompt window that was previously created.
2. Replaced `<otherhostcertificate.crt>` with the certificate file name being imported.
3. The alias is `icp<otherhostname>` where `<otherhostname>` is the hostname of the remote cluster node whose certificate the user is importing.
5.4.1 Deploying the Keystore File
The icp.keystore file in the workspace directory that now has all new certificates applied and must be deployed to the product before its contents can be used.

1. As a precaution, first rename the existing icp.keystore file located in the JBoss ICP configuration directory (<InstallDir>\jboss\icpconfig).
   a. Using Windows Explorer, right-click on the icp.keystore file and select Rename from the popup menu.
   b. Change the name (e.g., icp.keystore.YYYYMMDD), then press Enter.

Copy the icp.keystore file from the workspace directory to the JBoss ICP configuration directory.

Also copy all of the certificate files that were imported into the keystore to an alternate location (e.g., a certificates subdirectory in the JBoss ICP configuration directory or to another computer). This ensures that if any of the certificate files are somehow lost or damaged, you will not need to request the certificate(s) again.

5.5 Securing the JBoss Application Server (All nodes)

Note
This section is for enabling HTTPS for secure connectivity between a client web browser and the web server that runs in JBoss. This must be done in conjunction with enabling SSL for the Java Message Service communication between the Claim Connector and JBoss.

Once the signed certificates have been applied to the keystores, the system can be configured to support HTTPS.

Configuring a Standalone JBoss Instance for HTTPS
When CM/CES utilizes only a single server instance to implement its functions, it is termed a “standalone” system. In this case, JBoss configuration is performed by customizing the standalone.xml file in the <InstallDir>\jboss\standalone directory. Even in a multi-node configuration, each node still functions as an independent “standalone” system and therefore is configured by the same file.

Note
These instructions require modification of a configuration file that is critical to the operation of the JBoss application server. If you have any concerns, please contact Optum.

5.5.1 Configuring the Standalone.xml File
Configuration settings for Secure HTTP are normally commented out in the standalone.xml file. All changes to this file should follow the guidelines specified in section 2 - JBoss Configuration Procedures.

The “runtime” version of the standalone.xml file may be overwritten by JBoss and all comments removed from the file; therefore, it is important to always follow the guidelines to preserve your original and edited versions of the file.

Note
If the commented sections referred to in the following instructions are not found, refer to icp_standalone-default.xml file, a backup of the original standalone.xml file. Be aware that this will not have the latest configuration settings.
1. Stop all ICP services.

2. Using a text editor (such as Notepad), open the standalone.xml file located in the `<InstallDir>/jboss/icpConfig/standalone` directory.

3. Search for the `<system-properties>` element (near the top of the file), then locate the `useSSLClientAuthentication` property and change the value to `true`.

   **Original configuration**
   ```xml
   <property name="useSSLClientAuthentication" value="false"/>
   ```

   **New configuration**
   ```xml
   <property name="useSSLClientAuthentication" value="true"/>
   ```

   Locate the `localHostName` property and change its value from “localhost” to the hostname that was used in the Certificate Signing Request.

   **Original configuration**
   ```xml
   <property name="localHostName" value="localhost"/>
   ```

   **New configuration**
   ```xml
   <property name="localHostName" value="yourHostName"/>
   ```

   Locate the `keyAlias` property and change its value to the alias of the SSL digital certificate that was applied for this computer.

   **Original configuration**
   ```xml
   <property name="keyAlias" value="icplocalhost"/>
   ```

   **New configuration**
   ```xml
   <property name="keyAlias" value="yourCertificateAlias"/>
   ```

   Locate the `KeystorePassword` property and change its value to the correct value if you are not using the default password of "Ingenix1". (It is strongly recommended that this be changed.)

   **Original configuration**
   ```xml
   <property name="KeystorePassword" value="ingenix1"/>
   ```

   **New configuration**
   ```xml
   <property name="KeystorePassword" value="<new_password>"/>
   ```

   Locate the `TruststorePassword` property and change its value to the correct value if you are not using the default password of "Ingenix1". (It is strongly recommended that this be changed.)

   **Original configuration**
   ```xml
   <property name="TruststorePassword" value="ingenix1"/>
   ```

   **New configuration**
   ```xml
   <property name="TruststorePassword" value="<new_password>"/>
   ```

   Locate the `JMSport` property and change its value from 61616 to 61617.

   **Original configuration**
   ```xml
   <property name="JMSport" value="61616"/>
   ```

   **New configuration**
   ```xml
   <property name="JMSport" value="61617"/>
   ```
New configuration

```xml
<property name="JMSport" value="61617"/>
```

Locate the `<!--security-realm name="ManagementRealm">` element and remove the comments for the `<server-identities>` sub-element. (Delete the highlighted characters.)

**Original configuration**

```xml
<security-realm name="ManagementRealm">
  <server-identities>
    ... etc.
  </server-identities>
</security-realm>
```

**New configuration**

```xml
<security-realm name="ManagementRealm">
  <server-identities>
    ... etc.
  </server-identities>
</security-realm>
```

Locate the `<!--security-realm name="SSLRealm">` element and remove the comments for the block. (Delete the highlighted characters.)

**Original configuration**

```xml
<security-realm name="SSLRealm">
  ... etc.
</security-realm>
```

**New configuration**

```xml
<security-realm name="SSLRealm">
  ... etc.
</security-realm>
```

Locate the `<socket-binding>` element within the `<management-interfaces>` element. Comment out the entry for HTTP and remove the commenting for the HTTPS element as shown below.

**Original configuration**

```xml
<socket-binding http="management-http"/>
<socket-binding https="management-https"/>
```

**New configuration**

```xml
<socket-binding http="management-https"/>
<socket-binding https="management-https"/>
```

Locate the `connector-ref` elements. Comment out the entry for HTTP and remove the commenting for the HTTPS element as shown below.

**Original configuration (case 1)**

```xml
<remote connector-ref="http-remoting-connector" thread-pool-name="default"/>
<remote connector-ref="https-remoting-connector" thread-pool-name="default"/>
```

**New configuration (case 1)**

```xml
<remote connector-ref="https-remoting-connector" thread-pool-name="default"/>
```

© 2020 Optum, Inc.
Locate the `http-connector` elements. Comment out the entry for HTTP and remove the commenting for the HTTPS element as shown below.

**Original configuration**

```xml
<http-connector name="http-remoting-connector" connector-ref="default"
    security-realm="ApplicationRealm"/>
<!-- http-connector
    name="https-remoting-connector" connector-ref="https"
    security-realm="ApplicationRealm"/>
```

**New configuration**

```xml
<!-- http-connector
    name="http-remoting-connector" connector-ref="default"
    security-realm="ApplicationRealm"/>
<http-connector name="https-remoting-connector" connector-ref="https"
    security-realm="ApplicationRealm"/>
```

Locate the `http-listener` element. Comment out the entry for HTTP and remove the commenting for the HTTPS element as shown below.

**Original configuration**

```xml
<http-listener name="default" socket-binding="http" redirect-socket="https"
    enable-http2="true" max-post-size="1073741824" max-parameters="2000"
    no-request-timeout="1800000"/>
<!--https-listener name="https" socket-binding="https" enable-http2="true"
    security-realm="SSLRealm" verify-client="REQUESTED"
    enabled-protocols="TLSv1,TLSv1.1,TLSv1.2" max-post-size="1073741824"
    no-request-timeout="1800000" max-parameters="2000"/>
```

**New configuration**

```xml
<!--http-listener name="default" socket-binding="http" redirect-socket="https"
    enable-http2="true" max-post-size="1073741824" max-parameters="2000"
    no-request-timeout="1800000"/>
<https-listener name="https" socket-binding="https" enable-http2="true"
    security-realm="SSLRealm" verify-client="REQUESTED"
    enabled-protocols="TLSv1,TLSv1.1,TLSv1.2" max-post-size="1073741824"
    no-request-timeout="1800000" max-parameters="2000"/>
```

Locate the `session-cookie` element and change the value of the “secure” attribute from `false` to `true`.

**Original configuration**

```xml
<session-cookie http-only="true" secure="false"/>
```

**New configuration**

```xml
<session-cookie http-only="true" secure="true"/>
```

Locate the `socket-binding name="http"` element and comment it out as shown below.

**Original configuration**

```xml
<socket-binding name="http" port="${jboss.http.port:8080}"/>
```

**New configuration**

```xml
<!--socket-binding name="http" port="${jboss.http.port:8080}"-->
2. If you want to change the default HTTPS port to a custom value, locate the `<socket-binding name="https"` element and change the port value as shown below. (Port 8080 is used in this example, but any other available port can be used.)

**Original configuration**

```
<socket-binding name="http" port="${jboss.http.port:8443}"/>
```

**New configuration**

```
<socket-binding name="http" port="${jboss.http.port:8080}"/>
```

Alternatively, you can leave the above configuration unchanged and instead add a new parameter to the `conf\jboss.wrapper.conf` file as shown below.

```
wrapper.java.additional.18=-Dorg.tanukisoftware.wrapper.WrapperSimpleApp.waitForStartMain=TRUE
wrapper.java.additional.19=-Dorg.tanukisoftware.wrapper.WrapperSimpleApp.maxStartMainWait=60
wrapper.java.additional.20=-Djdk.tls.ephemeralDHKeySize=2048
wrapper.java.additional.21=-Djboss.https.port=8080
```

Save the `standalone.xml` file.

Copy the `standalone.xml` file from the `<InstallDir>`\`jboss\icpConfig\standalone` directory to the `<InstallDir>`\`jboss\standalone\configuration` directory.

### 5.5.2 Securing Remote EJB Access (`jboss-ejb-client.properties`)

Some command-line utilities (e.g., startSmartLoadKB.bat, LCDBulkLoader.bat, etc.) invoke EJB services that are running in the JBoss application server. Prior to the introduction of Wildfly (JBoss 8), this was done via Java Naming and Directory Interface (JNDI) lookups implemented via the JBoss JNP project. Since JBoss 8, the client side of the JNP project has been replace with the JBoss Remote Naming project, mostly to allow more fine-grained security configuration for JNDI lookups.

By default Wildfly uses 8080 as the remoting port for remote communication via HTTP. The objective of this section is to change the configuration to enable the use HTTPS and the associate TCP port number.

1. As a precaution, create a backup copy of the `jboss-ejb-client.properties` file in the `<InstallDir>`\`conf` directory.

Using a text editor (such as Notepad), open the `jboss-ejb-client.properties` file and change the `host` and `port` entries to the correct values.

| Note | Typically port 443 is used for HTTPS as opposed to 8080 for HTTP, but JBoss prefixes all standard ports with an additional digit to allow multiple instances of JBoss to run on the same computer without any port conflicts. Each instance is simply assigned a different prefix digit (“8” in this case). |

Uncomment the `remote.connection.default.protocol` property line by removing the “#” character from the front of the line. This enables HTTPS.

Change the property value from `false` to `true` on the last four lines.
Sample configuration

```properties
endpoint.name=client-endpoint
remote.connections=default
remote.connection.default.host=yourHostName
remote.connection.default.port=8443
remote.connection.default.protocol=https-remoting
remote.connection.default.connect.options.org.xnio.Options.SSL_ENABLED=true
remote.connection.default.connect.options.org.xnio.Options.SSL_STARTTLS=true
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS=true
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOPLAINTEXT=true
```

Save the new version of the file and exit the text editor.

### 5.5.2.1 Updating the Keystore Password in Batch Scripts

**Note**  As of 5.4 SP2-CU03, the manual update of batch scripts is no longer required. The scripts have been enhanced to automatically retrieve the required security configuration information from the configuration file for the ICP Service.

The batch scripts for loading the KnowledgeBase and for bulk load of LCD carrier data make remote EJB calls to services running in the JBoss application server to perform their function. When using HTTPS, they must access the certificate information in the keystore file in order to function properly. This requires that they have the correct password for accessing the keystore file. If the keystore password has been changed from its default value (recommended), then the password that is hardcoded in these scripts must be updated.

#### Configuration File Location

<table>
<thead>
<tr>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;InstallDir&gt;\bin\startSmartloadKB.bat</code></td>
<td><code>&lt;InstallDir&gt;\bin\startSmartloadKB.sh</code></td>
</tr>
<tr>
<td><code>&lt;InstallDir&gt;\bin\LCDBulkLoader.bat</code></td>
<td><code>&lt;InstallDir&gt;\bin\LCDBulkLoader.sh</code></td>
</tr>
</tbody>
</table>

1. Navigate to the `<InstallDir>\bin` directory. Perform the following for each of the two scripts identified above.
2. Open the script using a text editor (such as Notepad).
3. Locate the following group of four properties under the heading #These options are used when SSL_ENABLED is set to “true”. (Note that the Linux shell scripts are slightly different from the Windows batch script example shown below, but these lines should still be easily recognized.) Update the highlighted passwords with the correct plain text password.

```properties
:: These options are used when SSL_ENABLED is set to "true" in jboss-ejb-client.properties
set VMOPTIONS%$VMOPTIONS% -Djavax.net.ssl.keyStore=../jboss/icpConfig/icp.keystore
set VMOPTIONS%$VMOPTIONS% -Djavax.net.ssl.keyStorePassword=ingenix1
set VMOPTIONS%$VMOPTIONS% -Djavax.net.ssl.trustStore=../jboss/icpConfig/icp.keystore
set VMOPTIONS%$VMOPTIONS% -Djavax.net.ssl.trustStorePassword=ingenix1
```

4. When finished, save the edited file.

### 5.5.3 Configuring the ICP Service for HTTPS

The **ICP Service** (Bifrost) which has the responsibility of starting the other ICP services: **ICP Engine** (JBoss AS) and **ICP Connector**. When the ICP Engine is started, the **ICP Service** must recognize when it is available for use. It performs a check to test if the Web Service is available for browser connections. It must therefore be configured to connect to the Web Service via the HTTPS protocol rather than the default HTTP protocol. This section describes how to make this configuration change.
5.5.3.1 Configuring the services.properties File

1. As a precaution, first create a backup copy of the services.properties file in the <InstallDir>\conf directory.

2. Using a text editor (such as Notepad), open the services.properties file. Scroll to the bottom of the file to find the lines that match the properties in the following table.

3. Locate the following group of four properties under the heading #Security Properties. These properties use the non-secure values as a default (http) and must be changed to use secure values (https). Also replace localhost with your machine’s actual hostname (yourHostName).

   The following table lists the correct secure values.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>Secure Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>jboss.scheme</td>
<td>http</td>
<td>https</td>
</tr>
<tr>
<td>jboss.hostname</td>
<td>localhost</td>
<td>yourHostName</td>
</tr>
<tr>
<td>jboss.port</td>
<td>8080</td>
<td>8443</td>
</tr>
<tr>
<td>jboss.jindi.prefix</td>
<td>http-remoting://</td>
<td>https-remoting://</td>
</tr>
</tbody>
</table>

4. When finished, save the edited file.

5.5.3.2 Configuring the icp.wrapper.conf File

When using HTTPS, the ICP Service must also know the password for accessing the keystore file. If the default password (ingenix1) is changed, this must also be updated in the icp.wrapper.conf file.

1. As a precaution, first create a backup copy of the icp.wrapper.conf file in the <InstallDir>\conf directory.

2. Using a text editor (such as Notepad), open the icp.wrapper.conf file.

3. Locate the following group of four properties under the heading #Java Additional Parameters. Update the highlighted passwords with the correct plain text password

   ```
   #Java Additional Parameters
   wrapper.java.additional.10=-Djavax.net.ssl.keyStore=@INSTALL.DIR@/@JBOSS.DIR@/icpConfig/icp.keystore
   wrapper.java.additional.11=-Djavax.net.ssl.keyStorePassword=ingenix1
   wrapper.java.additional.12=-Djavax.net.ssl.trustStore=@INSTALL.DIR@/@JBOSS.DIR@/icpConfig/icp.keystore
   wrapper.java.additional.13=-Djavax.net.ssl.trustStorePassword=ingenix1
   ```

4. When finished, save the edited file.

5.5.4 Securing JMS Communication

Clients of CM/CES products send claims for editing over a TCP socket connection to the Claim Connector process. The Claim Connector then sends claims as “messages” via the Java Messaging Service (JMS) to the JMS server (message broker), which also runs in the same JVM as the Claim Connector. The various application server nodes (JBoss) also connect to the JMS Server and pull claims waiting to be analyzed from a message queue. When claim analysis is complete, a claim response message is returned back to the JMS Server which routes them back to the Claim Connector where they originated. Ultimately, the claim response is sent back to the client in the original encapsulation format (XML, IMF/EMF). If High Availability is enabled, there will be at least two JMS Server nodes (main and standby) that are clustered together. The
associated Claim Connectors must be configured to communicate with both JMS Servers over TCP with SSL encryption. In like manner, each JBoss application server must also be configured to communicate with both JMS Servers over TCP with SSL encryption. This is necessary to ensure that the Personal Health Information in claim data that must be legally protected is being sent over a network connection that is secure. This section describes how this JMS communication channel can be encrypted via digital certificates and thus made secure.

5.5.4.1 Configuring the connector.wrapper.conf File

When using HTTPS, the Connector Service must also know the password for accessing the keystore file. This is because the ActiveMQ/Artemis message broker (JMS provider) must communicate via HTTPS with each JBoss instance. If the default password (ingenix1) is changed, this must also be updated in the connector.wrapper.conf file.

1. As a precaution, first create a backup copy of the connector.wrapper.conf file in the <InstallDir>/conf directory.
2. Using a text editor (such as Notepad), open the connector.wrapper.conf file.
3. Locate the following group of four properties under the heading #Java Additional Parameters. Update the highlighted passwords with the correct plain text password.

```
#Java Additional Parameters
:  
wrapper.java.additional.10=-Djavax.net.ssl.keyStore=@INSTALL.DIR@/@JBOSS.DIR@/icpConfig/icp.keystore
wrapper.java.additional.11=-Djavax.net.ssl.keyStorePassword=ingenix1
wrapper.java.additional.12=-Djavax.net.ssl.trustStore=@INSTALL.DIR@/@JBOSS.DIR@/icpConfig/icp.keystore
wrapper.java.additional.13=-Djavax.net.ssl.trustStorePassword=ingenix1
```

4. When finished, save the edited file.

5.5.4.2 Configuring the broker.xml File

The broker.xml file is also located in the <InstallDir>/conf directory. Edit this file with a text editor to make the changes shown below.

1. Locate the <connectors> element that immediately follows the <ha-policy> elements. Uncomment the appropriate section for your configuration. Like the jndi.properties file, this section contains configurations for four different options:

- Single connector node without SSL (default).
- High Availability without SSL.
- Single connector node with SSL enabled.
- High Availability with SSL enabled.

   a. Uncomment the <connector> elements below the comment line for the appropriate option. Make sure that the <connector> elements for all other options are commented out with <!-- characters at the beginning of the line and --> characters at the end of the line.

   b. In the uncommented lines, make sure that the <mainNameOrIP> and <standbyNameOrIP> place holders are replaced with the actual hostname or IP address of the computer. The following example shows configuration for the “HA without SSL” option.
## Original configuration

```xml
<-- Single node without SSL settings (default) -->
<connector name="vm-connector">vm://0</connector>

<-- HA without SSL settings template -->
<connector name="main-connector">tcp://<mainNameOrIP>:61616</connector>
<connector name="standby-connector">tcp://<standbyNameOrIP>:61616</connector>
```

## New configuration

```xml
<-- Single node without SSL settings (default) -->
<connector name="vm-connector">vm://0</connector>

<-- HA without SSL settings template -->
<connector name="main-connector">tcp://MainHostname:61616</connector>
<connector name="standby-connector">tcp://StandbyHostname:61616</connector>
```

### Note

If more than one standby JMS server is needed, add a `<connector>` element for each standby node and name them “standby1-connector”, “standby2-connector”, etc.

2. Locate the `<acceptors>` element that immediately follows the `<connectors>` element. This section has two configurations: one for non-SSL and one for SSL.
   
a. If you are enabling SSL for either a single node or for multi-node high availability, comment out the two acceptors in the “Non-SSL” section (highlighted in YELLOW) and uncomment the acceptor in the “SSL settings template” section.

   Also, if you have changed the default location for your `keystore` and `truststore` files, you must update the values highlighted in GREEN below.

```xml
<acceptors>
  <!-- Non-SSL settings (default) -->
  <!--acceptor name="netty-acceptor">tcp://0.0.0.0:61616</acceptor-->
  <acceptor name="vm-acceptor">vm://0</acceptor-->

  <!-- SSL settings template -->
  <acceptor name="netty-acceptor">tcp://0.0.0.0:61617?directDeliver=false&sslEnabled=true;
  keyStorePath=C:/Optum/ICP/jboss/icpConfig/icp.keystore;keyStorePassword=ingenix;
  trustStorePath=C:/Optum/ICP/jboss/icpConfig/icp.keystore;trustStorePassword=ingenix;
  enabledProtocols=TLSv1,TLSv1.1,TLSv1.2
  </acceptor>
</acceptors>
```

b. You may also need to change the default passwords for the `keystore` and `truststore` files (highlighted in CYAN above).

3. Save the file and exit the editor.

### 5.5.4.3 Configuring the jndi.properties File

Located in the `<InstallDir>/conf` directory, the `jndi.properties` file has been pre-configured with four different configurations options:

- **Standalone (single node) without HTTPS/SSL** – This is the default “out-of-the-box” configuration and requires no changes.

- **High Availability without HTTPS/SSL enabled** – Only configuration for JMS server failover is required.
● **Standalone (single node) with HTTPS/SSL enabled** – Configuration changes for security plus application of SSL certificates is required.

● **High Availability with HTTPS/SSL enabled** – Configuration changes for security, application of SSL certificates, and JMS server failover must be configured.

1. After determining which of the above options you require, make a backup copy of `jndi.properties` and open it with a text editor.

2. Locate the `connectionFactory` line and uncomment it by removing the “#” from the front of the line.

3. Update this line by replacing the `<mainNameOrIP>` and `<standbyNameOrIP>` tags with the actual machine name or IP Address of the main and standby JMS server nodes.

4. Comment out all other `connectionFactory` lines by inserting '#' at the beginning of the line.

   **EXAMPLE:** HA with SSL

   **Original configuration** (HA with SSL)

   ```
   ### HA and ssl ###
   #connectionFactory.ConnectionFactory=(
   #    tcp://<mainNameOrIP>:61617
   #      ?sslEnabled=true,
   #      &enabledProtocols=TLSv1.2,
   #    tcp://<standbyNameOrIP>:61617
   #      ?sslEnabled=true,
   #      &enabledProtocols=TLSv1.2
   #  )
   #  ?consumerWindowSize=10240
   #  &minLargeMessageSize=409600
   #  &blockOnDurableSend=false
   #  &preAcknowledge=true
   #  &threadPoolMaxSize=-1
   #  &ha=true
   #  &retryInterval=1000
   #  &retryIntervalMultiplier=1.0
   #  &reconnectAttempts=-1
   ```

   **New configuration**

   ```
   ### HA and ssl ###
   connectionFactory.ConnectionFactory=(
   tcp://<mainNameOrIP>:61617
   ?sslEnabled=true,
   &enabledProtocols=TLSv1.2,
   tcp://<standbyNameOrIP>:61617
   ?sslEnabled=true,
   &enabledProtocols=TLSv1.2
   )
   ?consumerWindowSize=10240
   &minLargeMessageSize=409600
   &blockOnDurableSend=false
   &preAcknowledge=true
   &threadPoolMaxSize=-1
   &ha=true
   &retryInterval=1000
   &retryIntervalMultiplier=1.0
   &reconnectAttempts=-1
   ```

5. Save the file and exit the editor.

5.5.4.4 Updating the Keystore File for the Connector and JMS Server

The final step in the keystore configuration is to update the `truststore.ks` file with the public certificate for the machine on which the Claim Connector is running. This is required to allow the
Claim Connector to connect to the JMS server over the network when secure connections are enabled. Note that in a multi-node environment this step must be performed on each node where the Claim Connector will run. The certificate for that specific node must be imported into the truststore.ks file on that node.

| Note | The file name is truststore.ks, not keystore.ks. Both files will exist in the conf directory. |

1. Copy the truststore.ks file from <InstallDir>\conf to your workspace directory. (e.g., C:\IcpWorkspace).

Open a Command Prompt window and execute the following keytool command, replacing the <yourKeyAlias> and <CACertificateName> tokens with the correct values for this machine.

```
keytool -import -alias <yourKeyAlias> -file <CACertificateName>.crt -keystore truststore.ks
```

Copy the updated truststore.ks file from the workspace directory, back to the <InstallDir>\conf directory, overwriting the original file.

5.5.5 Configuring HTTPS with High Availability Enabled

This section covers configuring HTTPS in a system that has implemented JMS Server clustering (main and backup nodes), as far as exchanging public SSL certificates to allow connection.

The icp.keystore file is located in <ICP_Install_Dir>\jboss\icpconfig directory

The truststore.ks file is located in <ICP_Install_Dir>\conf directory.

- The icp.keystore files on each node must have their own key and certificate.
- The icp.keystore files on each node must also contain the public certificate for both the main JMS Server node and the standby JMS Server node. If HA is not enabled, meaning that only a single JMS Server node is running, then each icp.keystore (for each JBoss instance) must contain the public certificate for that single JMS Server node.
- The truststore.ks file on the JMS Server nodes must contain each other’s public certificates.

5.5.6 Configuring Rules Install for Secure HTTP (DB update)

| Note | This step will only need to be performed once when configuring a multi-node system. This is because the information is stored in the database which is shared by all nodes. Once you have enabled HTTPS, you only need to load a KB through the UI once on one of the nodes. |

These settings will define how the rules install connects to ILOG. Since the setting specifies a machine name, and it must be on the same machine from which the rules are being loaded, we suggest that in a clustered environment you use the master node machine name and remember to always load rules from the master node.

The values highlighted in yellow are what you are changing. You need to replace “YourCertHostname” with the name that correctly resolves to your master machine and which is allowed by your signed certificate. If you used a port other than 8443, you need to replace that value. Do not change “https.”

Using a database client, log in to the database and execute the following database script. The first script will display your current values. The second script will change the values.
Oracle

```sql
-- View existing configuration settings
select * from CONFIG_PROPERTY where PATH = '/t360/framework/rulesmanagement';

-- Update specific properties for rules install
update CONFIG_PROPERTY set PROPERTY_VALUE = 'YourCertHostname' WHERE PATH = '/t360/framework/rulesmanagement' AND PROPERTY_NAME='hostname';

update CONFIG_PROPERTY set PROPERTY_VALUE = '8443' WHERE PATH = '/t360/framework/rulesmanagement' AND PROPERTY_NAME='port';

update CONFIG_PROPERTY set PROPERTY_VALUE = 'https' WHERE PATH = '/t360/framework/rulesmanagement' AND PROPERTY_NAME='scheme';
commit;
```

SQL Server (assumes that the database name is “icp”)

```sql
-- View existing configuration settings
use icp;
select * from CONFIG_PROPERTY where PATH = '/t360/framework/rulesmanagement';

-- Update specific properties for rules install
use icp;
update CONFIG_PROPERTY set PROPERTY_VALUE = 'YourCertHostname' WHERE PATH = '/t360/framework/rulesmanagement' AND PROPERTY_NAME='hostname';

update CONFIG_PROPERTY set PROPERTY_VALUE = '8443' WHERE PATH = '/t360/framework/rulesmanagement' AND PROPERTY_NAME='port';

update CONFIG_PROPERTY set PROPERTY_VALUE = 'https' WHERE PATH = '/t360/framework/rulesmanagement' AND PROPERTY_NAME='scheme';
```

5.5.7 Restarting Services

On each node in the system, start the ICP services using the desktop shortcut and verify that the appropriate services were able to start cleanly. On nodes where the Claim Connector is configured to run, all three ICP services will be started (ICP Service, ICP Connector, and ICP Engine service). On other nodes, only the ICP Service and ICP Engine service will be started.

Note
---
From this point on, you will only be able to access the user interface for Claims Manager or Claims Edit System by using the HTTPS protocol (e.g., https://server:8443/cm/app).
6 Enabling ACE Reporting User Interface

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>This section only applies to 5.4 SP1 CU02+.</th>
</tr>
</thead>
</table>

1. Install ACE License.
2. Go to `<InstallDir>\conf` folder. Locate `ace.properties.template` file. Remove the `.template` extension and save it as `ace.properties`.
3. Go to the PE/FE enterprise and override the ACE Clinical Editing in EDI Stream to set ‘yes’ as value as shown below.

4. The ACE Reporting User Interface can be enabled from backend by executing a query in the DB –

   ```sql
   update CONFIG_PROPERTY set PROPERTY_VALUE = 'Y' where PROPERTY_NAME = 'AceReports';
   ```

5. Restart the ICP services.
Command-Line Tools

This chapter covers the command-line utilities that are delivered with CM/CES. Some are needed during the installation of the product (password encryption), but most are optional based on the needs of the end-user.

1 Load KnowledgeBase

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>startSmartLoadKB.bat</td>
<td>&lt;InstallDir&gt;\bin</td>
</tr>
<tr>
<td>startSmartLoadKB.sh (Linux)</td>
<td></td>
</tr>
</tbody>
</table>

**Command Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-full or -f</td>
<td>Load all data in the KnowledgeBase.</td>
</tr>
<tr>
<td>-update or -u</td>
<td>Load only tables where data has been updated since the last time the KnowledgeBase was loaded.</td>
</tr>
<tr>
<td>-help or -h</td>
<td>Prints usage information.</td>
</tr>
</tbody>
</table>

This command-line tool is available for clients that wish to avoid using the graphical user interface for loading new KnowledgeBase extracts as they become available. It allows a further level of automation as it can be executed remotely by some other controlling process.

**Running startSmartLoadKB**

1. Copy the KB Extract to be loaded to the `<InstallDir>\KB_Extracts` directory.

**Note** When using the graphical UI to load the KnowledgeBase, you can automatically upload the KB Extract from a location on your local workstation to the application server where it is then loaded into the database. However, the command-line tool requires that the KB Extract first be uploaded to a “known” location (the `KB_Extracts` directory).

2. Log in to the server that is running the CM/CES application.

3. Open a Command Prompt window. (This does not need to be opened “as Administrator”.)

4. Change directory to the `<InstallDir>\bin` directory.
   (Example: `cd C:\Optum\ICP\bin`)

5. Execute the startSmartLoadKB command.
   (Example: `.\startSmartLoadKB -u`)
2 Bulk Load LCD Carriers

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCDBulkLoader.bat</td>
<td>&lt;InstallDir&gt;\bin</td>
</tr>
<tr>
<td>LCDBulkLoader.sh (Linux)</td>
<td></td>
</tr>
</tbody>
</table>

### Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>Activate carriers after they are loaded.</td>
</tr>
<tr>
<td>no-activate</td>
<td>Do not activate carriers after they are loaded.</td>
</tr>
<tr>
<td>No parameter or invalid parameter</td>
<td>Normally treats this the same as the <code>activate</code> option. However, the script can be edited to change the <code>ACTIVATE_LCD_CARRIERS</code> variable to from <code>TRUE</code> to <code>FALSE</code>, in which case the default action is to not activate carriers after they are loaded.</td>
</tr>
</tbody>
</table>

This command-line tool is available for clients that wish to avoid using the graphical user interface for loading multiple LCD Carrier files as a batch. It allows a further level of automation as it can be executed remotely by some other controlling process.

1. After the installation of 5.4 SP2 CU01, the default action of LCDbulkloader.bat is to both load and activate all LCD carriers in the LCD_Extracts directory. Prior to loading this CU, carriers are loaded, but are not activated.

2. ICP services must be running at the time that the LCDBulkLoader is used. It is not necessary to stop claim processing during the loading and activation of LCD Carriers. As soon as the data load and activation are complete, claims received from that point on will use the new data.

### Running LCDBulkLoader

1. Copy the LCD Carrier files to be loaded to the `<InstallDir>\LCD_Extracts` directory.

   **Note** When using the graphical UI to load LCD Carrier data, you can upload LCD Carrier data files from a location on your local workstation to the application server where it is then loaded into the database. However, the command-line tool requires that the LCD Carrier data file first be uploaded to a “known” location (the `LCD_Extracts` directory).

2. Log on to the server that is running the CM/CES application. Note that ICP Services must be running in order to use this tool.

3. Open a Command Prompt window. (This does not need to be opened “as Administrator”.)

4. Change directory to the `<InstallDir>\bin` directory.
   (Example: `cd C:\Optum\ICP\bin`)

5. Execute the `LCDBulkLoader` command. Note that if no parameters are specified, the carriers will be activated after they are loaded.
   (Example: `LCDBulkLoader` or `LCDBulkLoader no-activate`)

6. A log file (`LCDBulkLoader.log`) for LCDBulkLoader actions is updated in the `<InstallDir>\log` directory.
7. After an LCD Carrier file in the LCD_Extracts folder is loaded into the database, the file is moved to the <InstallDir>\LCD_Extracts\processed directory.

3 Password Encryption

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>encjboss.bat</td>
<td>&lt;InstallDir&gt;\bin</td>
</tr>
<tr>
<td>encjboss.sh (Linux)</td>
<td>&lt;InstallDir&gt;\bin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>The only parameter is the plain text password that is to be encrypted.</td>
</tr>
</tbody>
</table>

To perform queries against the database, the application server (JBoss/Wildfly) must know the encrypted password for the product database user (icp_p). This encrypted password is automatically generated when the software is installed, but must be manually generated using the encjboss command whenever it needs to be changed in accordance with corporate password security policies. This script calls a JBoss component to encrypt the specified plaintext password and return it as a text value that can be copied into the standalone.xml configuration file. (Refer to section 3.6.2 - Password Encryption for further details.)

Running startSmartLoadKB

1. Log in to the server that is running the CM/CES application.
2. Open a Command Prompt window. (This does not need to be opened “as Administrator”.)
3. Change directory to the <InstallDir>\bin directory.
   (Example: cd C:\Optum\ICP\bin)
4. Execute the jbossenc command.
   (Example: \encjboss new_passwd
   Output: Encoded password: 190086d21f07a929dd5e67b9d75dab4e )
# 4 Configuration Saver

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>saver.bat</td>
<td>The Configuration Extractor Loader tool is typically installed on a user workstation rather than on a CM/CES application server. It can be installed in any directory chosen by the user.</td>
</tr>
</tbody>
</table>

## Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dbtype=</td>
<td>Oracle or MSSQL</td>
</tr>
<tr>
<td>--dburl=</td>
<td>The JDBC connection string to connect to the database. Example: jdbc:jtds:sqlserver://localhost:1433/ICP</td>
</tr>
<tr>
<td>--user=</td>
<td>The database account name to use when connecting to the database. Example: icp_p</td>
</tr>
<tr>
<td>--pass=</td>
<td>The encrypted password for the database user. This can only be obtained by using the Copy Parameters button of the Saver tool.</td>
</tr>
<tr>
<td>--file=</td>
<td>The full path to where the configuration backup data should be saved. Note that the file extension must be .h2.db.gz</td>
</tr>
<tr>
<td>--selections=</td>
<td>Enter a comma-separated list of the following backup options. The order is not significant, but there can be no spaces in the list. OVERRIDES RULES_AND_CONFIGURATION USERS KB LCD PQRS CLAIMS NPT_HISTORY FREQUENCY_HISTORY</td>
</tr>
</tbody>
</table>

## Return Values (%ERRORLEVEL% environment variable)

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The configuration backup was successfully created.</td>
</tr>
<tr>
<td>non-zero</td>
<td>An error occurred and the configuration backup cannot be guaranteed to have valid/complete configuration data.</td>
</tr>
</tbody>
</table>

Please refer to section 2.5.1 - Capturing a Configuration (command-line) for full details regarding command-line usage of this tool.
5 Configuration Loader

Command Name | Location
--- | ---
loader.bat | The Configuration Extractor Loader tool is typically installed on a user workstation rather than on a CM/CES application server. It can be installed in any directory chosen by the user.

Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dbtype=</td>
<td>Oracle or MSSQL</td>
</tr>
<tr>
<td>--dburl=</td>
<td>The JDBC connection string to connect to the database. Example: <code>jdbc:jtds:sqlserver://localhost:1433/ICP</code></td>
</tr>
<tr>
<td>--user=</td>
<td>The database account name to use when connecting to the database. Example: <code>icp_p</code></td>
</tr>
<tr>
<td>--pass=</td>
<td>The encrypted password for the database user. This can only be obtained by using the Copy Parameters button of the Loader tool.</td>
</tr>
<tr>
<td>--file=</td>
<td>The full path of the configuration backup data that is to be loaded.</td>
</tr>
</tbody>
</table>

Return Values (%ERRORLEVEL% environment variable)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The configuration load was successful.</td>
</tr>
<tr>
<td>non-zero</td>
<td>An error occurred and the loaded configuration data cannot be guaranteed to be valid/complete.</td>
</tr>
</tbody>
</table>

Please refer to section 2.5.2 - Loading a Configuration (command-line) for full details regarding command-line usage of this tool.
Support Tools

1 DDR Trace Instructions

1.1 Requirements

<table>
<thead>
<tr>
<th>Version</th>
<th>Prerequisite</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.1+ Claims Manager or 5.2.1+ Claims Edit System</td>
<td>KnowledgeBase Feb 2016+</td>
<td>Internet Explorer 10+</td>
</tr>
</tbody>
</table>

This works on the server or the end user’s computer that is not on server.

1.2 To Run a Trace within DDR

1. Log in to the product.
2. Open a new tab.
3. Enter this link:
   - http://localhost:8080/ices/ui/trace (for CES)
   - http://localhost:8080/cm/ui/trace (for CM)

   **Note**
   - If utilizing trace from the server then use localhost.
   - If utilizing trace via a browser not on the server, replace localhost with the machine name.
   - If trace opens the login screen, log in using your existing username and password.

4. Along the top there are three tabs: Available trace data, Trace filters and Import Trace file. Select Trace filters.
5. Click on the line for the pattern you want to trace or select Trace all. Most users will only trace one or a few patterns.

   **Note**
   - 6. Optum recommends not utilizing Trace all for an extended processing period, as it could impact processing performance.

6. Free text search options are available (Pattern ID, Type, Flag, Name, Trace).
7. Click the line of the pattern to trace, this activates trace for the pattern ID selected, the trace field turns to Yes.
8. To begin getting trace data, analyze a claim where you expect the flag to fire.
9. Select Available trace data. The claim that was analyzed will display.
10. Select the claim ID where you wish to view the trace. Select the line; this opens the trace data with all available lines.
11. Expand each line and click on the right arrow to view the information under the line.
7. Expand all possible right arrows to display all trace data.

12. To export the data, select **Export** in the top right corner and save it.

**Note** 8. The file can now be emailed or attached.

14. When finished, turn off the *Trace for patterns* by selecting the pattern line, in *Trace filters*. The trace column should change to **No**, or you can select **Untrace all**.

15. Close the browser.

### 1.3 Reading Available Trace Data

Arrows will display that will allow you to expand the data for pattern, claim, and line details. Expand all right arrows to view all trace data on a claim.
2 Configuration Capture and Loader

2.1 Overview

The Configuration Capture and Loader tool can be used to copy a full or partial CM/CES configuration from one machine to another. Usage examples include:

- Replicating a configuration from a client with issues so that Support, QA, and Development can do a detailed examination of a problem.
- Creating an identically configured TEST environment from an existing production environment to test a release upgrade or a configuration change prior to putting them into production.
- Backing up an environment’s configuration before doing configuration changes and using the backup as a way to “get back” to the configuration that was there previously.
- Switch the active configuration on the same machine without a new machine or install.

A configuration capture from one multi-tenant system can be loaded onto another multi-tenant system, but cannot be successfully loaded onto a single-tenant system. Additionally, a configuration capture from a single-tenant system cannot be successfully loaded onto a Multi-tenant system. An alert message is displayed when attempting to load an incompatible configuration capture.

2.2 Requirements and Methods of Running the Tool

<table>
<thead>
<tr>
<th>URL Method/Support page</th>
<th>Loader and Extractor Application</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 2016 Regulatory KB or later</td>
<td>February 2016 KB or later</td>
<td>Internet Explorer 11+</td>
</tr>
</tbody>
</table>

URL Method/Support page: Typically used by support or the client to obtain custom configuration data such as rulesets, routes and overrides. This method intentionally does not capture data that can be loaded in other ways, such as KB and LCD data. On the support page there is also a mechanism for obtaining de-identified claims. The claims are provided as XML claims that can be imported in the system and are stored separately from the configuration capture. No software is required and the URL is included with the product.

Loader and Extractor Application: The Configuration Capture and Loader application must be downloaded from Optum.com. Two applications can be run when this tool is obtained:

- Loader.bat: Used to load a previously extracted configuration on to an installed ICP machine. The loader will make broad changes to configuration and table data on the target machine and care should be taken when using this tool. See the Common Use Cases section to understand best practices.
- Saver.bat: Provides the ability to capture configuration like the URL/support page but includes additional options such as the ability to extract claims and KB data. No changes are made to the source system when using the save tool via the support page or saver.bat.
Detailed instructions for using each method are included as follows.

2.3 URL/Support Page Method

Available back to July KB 2016.

To use this method, the user must have access to the professional main enterprise within the application.

1. Log in to Claims Manager or Claims Edit System.

2. Use the following URL and place it in a new browser:
   - For CES – http://clientshostname:8080/ices/ui/ddr/support.html
   - For CM – http://clientshostname:8080/cm/ui/ddr/support.html

   **Note** An easy way to also use the URL is to remove /app/ and everything after it on any URL in ICP and add /ui/ddr/support.html.

3. The following window is displayed.
4. Select the appropriate options, as outlined in the table below.

5. Select Download. The file will be prepared on the ICP machine and made available for download.

6. The zip file will appear in your Downloads folder as “BACKUP_Product_CurrentDate_time.zip”.

7. The file can then be delivered to Support.

<table>
<thead>
<tr>
<th>Selections in Configuration Capture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capture Database</strong></td>
</tr>
<tr>
<td>System Configuration</td>
</tr>
<tr>
<td>This option captures data from your system's database that can allow the recreation of your system's environment on another system by using the loader component of the Configuration Capture tool. Creating the capture does not introduce significant load to the system and can be run on a production system.</td>
</tr>
<tr>
<td>This capture will contain the following data from the current system:</td>
</tr>
<tr>
<td>• ILOG rulesets and rules</td>
</tr>
<tr>
<td>• DDR rulesets and rules</td>
</tr>
<tr>
<td>• All overrides</td>
</tr>
<tr>
<td>• Enterprises and routes</td>
</tr>
<tr>
<td>• PQRS and LCD configuration data</td>
</tr>
<tr>
<td>Regarding data collected, this option is identical to the saver component of the Configuration Capture Tool's Backup Options when selecting RULES AND CONFIGURATION and OVERRIDES. It will not gather any data about system users or any claim data.</td>
</tr>
<tr>
<td>The captured configuration enables Optum support to replicate an environment aiding in identifying and resolving issues as quickly as possible.</td>
</tr>
<tr>
<td>The captured file will appear as CONFIG.h2.db inside the downloaded zip file.</td>
</tr>
</tbody>
</table>

| **Capture database structure information without any data** |
| This option captures data about the database schema on the current system, including: |
| • Tables & Columns                 |
| • Constraints                      |
| • Indexes                         |
| • Foreign Keys                     |
| • Triggers, Functions and Stored Procedures |
| The resulting zip file can then be uploaded to a different installation using the [ui/ddr/dbcompare.html page](#) where the above elements will be compared to those on the target system. In the resulting comparison, this installation will be referred to as source while the other system will be referred to as target. This mechanism is useful when performing a database health check or verifying a Cumulative Update or expected change to the database was correctly applied. |
| The captured file will appear as SCHEMA.h2.db inside the download zip file. |
| The resulting file does not include any protected health information (PHI), as defined by US federal law. |
## Selections in Configuration Capture

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include JBoss logs</td>
<td>This option gathers all the jboss.server.log* and the jboss.wrapper.log.* files to the log folder within the resulting .zip file. When this option is checked, it is possible that the resulting .zip file could contain elements of PHI.</td>
</tr>
<tr>
<td>Include connector logs</td>
<td>When this option is checked, it is possible that the resulting zip file could contain elements of PHI.</td>
</tr>
<tr>
<td>Include KB Load logs</td>
<td>This option gathers all the bifrost.log* and the icp.wrapper.log.* files to the &quot;log&quot; folder within the resulting .zip file.</td>
</tr>
</tbody>
</table>
| Include configuration files used to configure the connector | The following files will be gathered and added to the resulting .zip file:  
  - conf/facets.accountmap.properties  
  - conf/facets.errormap.properties  
  - conf/facets.grouptoroute.properties  
  - conf/facets.grouptoroute-fe.properties  
  - conf/facets.planmap.properties  
  - conf/facets-user-map.xml  
  - conf/messageadapters.xml  
  - conf/X12Mapping.properties  
  - conf/connector.wrapper.conf  
  - conf/broker.xml *(only for 5.4 SP1 and later)* |
| Include configuration files used to configure application server | The following files will be gathered and added to the resulting .zip file:  
  - conf/smartLoadMapping.xml  
  - conf/jboss.wrapper.conf  
  - On 5.4 systems  
    - jboss/standalone/configuration/standalone.xml  
    - jboss/icpConfig/standalone/standalone.xml  
  - On pre-5.4 systems  
    - jboss/server/apollo/conf/jboss-service.xml  
    - jboss/server/apollo/conf/log4j.xml  
    - jboss/server/apollo/conf/login-config.xml  
    - jboss/server/apollo/conf/standardjboss.xml  
    - jboss/server/apollo/deploy/hornetq/hornetq-configuration.xml  
    - jboss/server/apollo/deploy/hornetq/hornetq-jms.xml  
    - jboss/server/apollo/deploy/icp-cache.sar/jgroups.properties  
    - jboss/server/apollo/deploy/icp-cache.sar/jgroups.xml  
    - jboss/server/apollo/deploy/jbossweb-tomcat55.sar/server.xml  
    - jboss/server/apollo/deploy/jbossweb-tomcat55.sar/conf/web.xml  
    - jboss/server/apollo/deploy/apollo-ds.xml |
### Selections in Configuration Capture

This option will extract one or more claims from the current system specified by a list of claim IDs entered in a text box. The claims and their patient history and NPT history (depending on the history selection options) will be extracted as XMLv2 format, with the NPT history coming as a loadable CSV. This allows an example claim to be safely sent without PHI to Optum to aid in addressing a specific issue or question. The de-identification procedure for claims is as follows:

1. Patient information (First Name, Last Name, and Address excepting Zip Code) is never put into the exported claim.
2. Claim ID is generated uniquely for the set of claims being exported.
3. All patient history for a given Patient MRN is read and reassigned a different patient MRN that is generated for the patient.
4. Patient DOB is changed such that the patient age will transform as follows:
   - Patients with ages 0-1 yrs. become 6 months old
   - Ages 1-5 yrs. become 3 yrs. old
   - Ages 6-18 yrs. become 12 yrs. old
   - Ages 19-64 yrs. become 40 yrs. old
   - Ages 65 yrs. and over become 75 yrs. old
5. All claim dates for each patient (BDOS, EDOS, admission date, from date, through date, span dates, etc.) are randomly shifted by the same amount (up to 1 year's worth of days). This value is randomly selected for each patient. Thus the calendar distance between medical events remains the same for each patient but the actual dates are eliminated.
   - The patient ZIP is included so that jurisdiction based mappings that use zip codes can function, but if the zip code starts with one of the following 3-digit beginnings, those three digits are set to 000: "022", "036", "059", "102", "203", "555", "692", "821", "823", "830", "831", "878", "879", "884", "893", "987", "994".
   - (These are zip codes where the population is small enough that it could lead to a narrowing of possibilities).
6. The account EID is hard coded to "ACT", so the account is lost.

### Include specific de-identified claims (specified by current claim ID)

Include specific de-identified claims

---

### Include thread dump

**Do not select** unless requested from Optum Support.

This option gathers the current state of threads on the system. This can be vital for troubleshooting if there is a "hang" situation on the system.
### Selections in Configuration Capture

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include heap dump</td>
<td>Dumps the heap memory of the application to a file. This is useful for debugging memory leaks, such as when unneeded object references are retained indefinitely and not allowed to be garbage collected.</td>
</tr>
<tr>
<td>• Collecting this information can take time and add substantial size to the downloaded package. This should only be done when the engineering or support team at Optum recommends it.</td>
<td></td>
</tr>
<tr>
<td>• Systems that don't have a patching CU will need <code>&lt;path name=&quot;sun/management&quot;/&gt;</code> added to the module.xml in C:\Optum\ICP\jboss\modules\system\layers\base\sun\jdk\main to allow the Support page to access the necessary mechanism to trigger the thread dump.</td>
<td></td>
</tr>
<tr>
<td>When this option is checked, the resulting zip file could contain elements of PHI.</td>
<td></td>
</tr>
<tr>
<td>Include DDR license report</td>
<td>If there is a problem with pattern licensing, this report will provide information about the DDR rules the system has and their license status.</td>
</tr>
<tr>
<td>Include DDR cache diagnostics</td>
<td>Cache diagnostics provides a summary of the DDR rules and their health status on the system. If a rule fails to compile or has other problems, the failure will show up in cache diagnostics.</td>
</tr>
</tbody>
</table>

**Note**
For detailed information on the selections above, log into the support page by accessing the URL above and select the information icon at the end of each selection.

### 2.4 Configuration Saver and Loader Tools

The configuration **Saver** and **Loader** applications provide the ability to capture system configuration data in one environment and load it into a different environment.

The tool is available on the client portal by searching for “Configuration Extractor Loader Tool.” A build number appears in the title bar of the application. This build number will be referenced in any communications regarding changes and updates to the application.

The contents of the directory can be simply copied to the machine where it is intended to run. It does not need to be copied to either the CM/CES application server or the database server. It can be run from your local workstation if desired.

The following batch command files to execute the tools are provided:

- **saver.bat** Extracts the configuration data from a CM/CES system and saves it to a file on the machine from which the tool is executed.
- **loader.bat** Loads previously saved configuration data onto a different CM/CES system. Note that the saved configuration must be accessible on the machine from which the loader tool is executed.
- **claimcopy.bat** This is a combination of the above two tools that is used to copy all claims from a source database to a different target database. (See section **Claim Migration Tool** below for full details.)
1. The Saver and Loader tools may be run using a graphical interface (recommended) or with all required parameters being specified on the command-line. Both of these options are documented below.

2. A date-stamped log file is created as these tools run. The log file will provide details for any errors that occur.

### 2.4.1 Recommendations and Requirements

1. A network connection to the database of the system where loading or saving is required. (Special software such as the Oracle client is not required.)
2. Ensure services are shut off when performing the load.
3. It is recommended that the same product version be installed on both source and target machines. (The tool will prompt you if the versions are different.)
4. The tool can be used to migrate data from one database type to another (i.e., MS SQL Server to Oracle and vice versa).

### 2.4.2 Capturing a Configuration

**Note** It is highly recommended that the saver.bat tool be used to save a backup of the configuration before using the loader.bat to load a new configuration.

To run the configuration saver, navigate to the directory where the tool was installed using Windows Explorer and then double-click the saver.bat batch command file. The window shown at left appears with default values in most fields.

**Connection Configuration**

1. Select your **Database Type** (Oracle or MSSQL). The **Database URL** should appear by default as well as the **User** (icp_p) and an empty **Password** field. Note that the **Database URL** field will change depending on the type of database that was selected. If the database server for the CM/CES system whose configuration is to be captured is not the local machine, change “localhost” to the DNS hostname of the remote database server.

**Note** Remote Desktop access to either the CM/CES server or the database server is not required. Obtaining a configuration requires only access to the network the database is connected to and a valid account on the database.
2. The **Backup File** field populates with a default destination. A new location can be specified by clicking on the “…” icon to open a file browser to select a new path.

3. Click the **Test Connection** button to verify that your connection settings are valid. If everything is correct, a success dialog box will pop up. If not, an error dialog is displayed.

4. The **View Selected File** button is enabled when the backup has completed. This allows viewing the contents of the extract.

**BACKUP OPTIONS**

The Backup Options panel allows specification of which configuration data should be included in the backup file. Note that some of the data (KB, LCD, PQRS) can be loaded from other sources and normally should not be included.

5. The configuration **Saver** tool provides some features not available using the URL/Support page. These are noted below.

   a) The **RULES AND CONFIGURATION** and **OVERRIDES** are options identical to the behavior of the extracts produced by the URL method. The other settings are additional types of data that can be extracted.

   b) The **KB** and **LCD** options collect KnowledgeBase and LCD data that could otherwise be loaded from regularly released extracts; therefore, it is typically best to leave these option unchecked as they significantly increase the time to capture the configuration and the size of the backup file. Note that KB and LCD overrides are collected via the **OVERRIDES** option, not through these options.
c) The **USERS** and **CLAIMS** options back up data that could contain PHI. The **USERS** option will copy tables with names and accounts of all users on the ICP system. Password information is encrypted. This option is useful when creating a test environment or another production environment from an existing installation as it allows current users on the production system to also be able to access the test system.

The **CLAIMS** option will save data from all of the claim-related tables. This option is intended to be used when it is necessary to create a test environment from a production environment and a subset of the claims in the production environment is needed for testing.

<table>
<thead>
<tr>
<th>Important!</th>
<th>Selecting CLAIMS will extract PHI if the system contains real (production) claims (as opposed to test claims). The interim format does not perform encryption of this data.</th>
</tr>
</thead>
</table>

d) When the **CLAIMS** option is selected, the **Specify Date Range** fields are displayed. The **Last Analysis Date** on the claim must fall within the specified range. For example, retrieving claims for the last 90 days, claims analyzed during the last 90 days are saved.

e) When the **NPT HISTORY** option is selected, the data from the NPT History table will be included in the configuration capture. If checkboxes for both the **CLAIMS** option and its associated **Exclude Claims from the following enterprise IDs** option have been selected, then only the NPT history data for the same enterprises for which claims were extracted will be included.

f) When the **FREQUENCY HISTORY** option is selected, the data from the Frequency History table will be included in the configuration capture. If checkboxes for both the **CLAIMS** option and its associated **Exclude Claims from the following enterprise IDs** option have been selected, then only the frequency history data for the same enterprises for which claims were extracted will be included.

g) When the **REPORTS** option is selected, Classic reports and panel-based Reports will be included in the configuration capture.

| Note | Classic scheduled reports will not automatically run as scheduled reports. Once a configuration capture is loaded on a server, in order for scheduled reports to run the report must be edited/saved and then it will be scheduled. |
6. Click **Save** to start the backup. As the data is extracted, a progress bar appears as well as information about the number of rows extracted from each database table. The data is saved to a file that can later be used to load configuration onto another target database.

7. When the data extraction is complete, click **OK** to close the progress dialog. A `.db.gz` file is produced by the extractor which contains the captured configuration data for the selected options. The `.db.gz` file may then be moved to an archive location from which it can be later used to load into the database of a different system.

### 2.4.3 Loading a Saved Configuration

| **Note** | The load makes substantial changes to the target database. It is highly recommended that you use the saver.bat to perform a backup of the configuration prior to using the loader.bat to load a new configuration. |

1. Stop ICP services on the target machine before loading the database changes.
2. Copy or move the saved extract file to a machine with the loader and network access to the machine you wish to load.
3. Double-click the **loader.bat** batch command to display the screen shown below.
4. Reference Capturing a Configuration using the saver.bat section for information on populating Database URL, User and Password. Selecting a valid Backup File will enable the View Selected File button.

5. Upon clicking View Selected File, a console appears allowing SQL queries against the database in the extract. This screen can be used to verify data prior to loading. This is very useful to support or development in troubleshooting the load process.

6. To run the load, navigate back to the Configuration Loader window and select Load.

| Note | Prior to the load, several safety checks are performed by comparing the data to be loaded against the data in the target system. These safety checks allow you to cancel the load and leave the target machine unaltered. |

Safety Check Examples:

- **Different CU or Product Version between source and target machine:** The following message will appear if the Cumulative Update level or Product Version is different between your source and target system.

- **KnowledgeBase with different table structure on source and target:** The following screen appears when the configuration has a different KnowledgeBase and table structure. Tables identified will differ between source and target KB versions.
• **Different enterprises on source and target machine:** The following screen appears when the source and target systems have differences in their enterprise configuration.

![Decision required for claims on target system]

The target system contains claims, but the enterprise structure in the source system is different

Because of the difference in enterprise structure, if the load were to simply continue and update the enterprise structure on the target system the existing claims would be left in an enterprise that would no longer be valid for claim processing.

If the target system is a production version of a test system that has had enterprise configuration changes it is likely that making the same enterprise changes to the production system that were made in test will resolve this conflict and allow this configuration to be loaded.

Currently, the only options in the loader is to truncate all existing claims or cancel the load.

  - [Purge Claims]
  - [Cancel]

• **Target machine does not have LCD carriers loaded:** The following screen appears when the source configuration has routes that make use of LCD but the target system does not have the LCD carriers loaded. The LCD carriers can be loaded before or after the saved configuration is loaded, but be sure to make note of the listed carriers and to load them.

![Routes in the extract reference LCD data for the following contractors that is not present in the target system. You must load this data before running any claims against these routes to edit the same as the source system.]

National Coverage Determination - Part B (5) DATE:20160410
CSC Administrators, LLC (15102) DATE:20160425
Wisconsin Physicians Service Insurance Corporation (5102) DATE:20160425

  - [Continue]
  - [Cancel]

• **Target machine does not have PQRS data loaded:** The following screen indicates that the target system does not have PQRS data, but the source system had PQRS data loaded. Failure to load PQRS data could result in different editing results. PQRS data can be loaded before or after the saved configuration is loaded.

![You need PQRS Data]

The Source system had PQRS data loaded, but the target system does not.
You'll need that loaded if PQRS is to be used for correct editing.

You should be able to load PQRS data before or after the configuration load.

  - [Continue]
  - [Cancel]
7. A “Load Progress” dialog is displayed as the data load occurs. When complete, click OK. You can now start ICP services again.

![Load Progress dialog](image)

### 2.5 Configuration Capture/Load Via the Command-Line

In some scenarios a user may want to use the command line interface if they want to load and/or save a configuration from ICP as part of a larger automated process. Typically these processes involve the use of a script or job scheduler, such as Windows Task Scheduler or other third party products that may involve a monitoring system to page administrators if a load fails. Outside of these use cases, users are advised to use the user interface method for its simplicity and usability.

**Notes**

1. When initially setting up the ability to capture and save a configuration via the command-line, the `saver.bat` and `loader.bat` tools can be used to generate the desired command parameters which can then be inserted into a batch command script that will be used later. This is for convenience and to avoid input errors.

2. These tools set the `ERRORLEVEL` environment variable to indicate success or failure. This variable should be checked by scripts that call these tools so that any errors can be properly reported or handled. A non-zero value indicates failure.

#### 2.5.1 Capturing a Configuration (command-line)

The following assumes that the Configuration Extractor Loader Tool has already been downloaded from the customer portal as described in section 2.4 - Configuration Saver and Loader Tools above.

1. Follow the steps outlined in section 2.4.2 - Capturing a Configuration above to configure/test the database connection and specify the desired backup options. Do not click the Save
button to initiate the configuration capture. Instead, click **Copy parameters**. This will copy the parameters to the *Windows Clipboard*.

2. Open the application that will be running *saver.bat*. This could be the text editor where you create a batch command or shell script or the Windows *Task Scheduler, Start a Program* dialog box.
   a. Enter the full path to *saver.bat*, followed by a SPACE character.
      (e.g. "C:\Optum\ICP\Configloader\saver.bat")
   b. Paste the parameters that were saved in the *Windows Clipboard*. (e.g., Type *CNTL+V* or right-click with the mouse in the case of a Windows *Command Prompt* window.)

| Note | A detailed list of the parameters used by the **Saver** tool can be found in section 4 - **Configuration Saver** of the *Command-Line Tools* chapter. |

3. Save the batch command script or complete creation of the scheduler task. Exit from the editing application.

4. The newly created command or task can then be executed to run the *saver.bat* tool. Configuration backup files will be created in the directory where the tool is installed.

| Note | If using the *saver.bat* tool and CLAIMS have been selected, we recommend not analyzing claims while the configuration is saving. |

### 2.5.2 Loading a Configuration (command-line)

1. Follow the steps outlined in section 2.4.3 - **Loading a Saved Configuration** to configure/test the database connection to the target system. Do not click **Load** to initiate the configuration load. Instead, click **Copy parameters**. This will copy the parameters to the *Windows Clipboard*.

2. Open the script or application that will be running *loader.bat*. This could be the text editor where you create a batch command or shell script or the Windows *Task Scheduler, Start a Program* dialog box.
   a. Enter the full path to *loader.bat*, followed by a SPACE character.
      (e.g. "C:\Optum\ICP\Configloader\loader.bat")
   b. Paste the parameters that were saved in the *Windows Clipboard*. (e.g., Type *CNTL+V* or right-click with the mouse in the case of a Windows *Command Prompt* window.)

| Note | A detailed list of the parameters used by the **Saver** tool can be found in section 5 - **Configuration Loader** of the *Command-Line Tools* chapter. |

3. Save the batch command script or complete creation of the scheduler task. Exit from the editing application.

4. The newly created command or task can then be executed to run the *loader.bat* tool. Configuration backup files will be created in the directory where the tool is installed.

| Note | Remember that ICP Services must be stopped before running the batch script or scheduled task that executes the *loader.bat* tool. |
2.6 Common Use Cases

2.6.1 Creating a test environment from a production environment

It is often required to have a test environment that functions the same as a production environment to verify behavior or changes without affecting production. This section details how to create a test environment from an existing production environment by using the loader and extractor applications.

The pre-requisite to creating the test environment is to get the application installed.

Ideally, you will load the same KB on the test environment that is loaded on the production environment. An ILOG Rules Install is not required as this will be done via the load from the production environment.

On the initial first configuration load, when initializing creating a test environment from a production environment, it may be desirable to add the “USERS” option when running the saver.bat.

Best practice guidelines summary:

- Installed ICP system with same version of the application
- Load the same KB on test as on the production environment
- Load LCD and PQRS data if the production system is using
- Perform a configuration capture of the production environment.

Method 1: Use the saver.bat to capture configuration from the production environment.

   **Advantage:** You can select USERS. This will cause ALL production users to be set up in your test environment with their same credentials and access rights.

   **Disadvantage:** You must have database access to the production environment to do this operation. For production machines, these rights are commonly held by different people than those administering test systems.

Method 2: Use the support page on the Web UI to capture configuration from the production environment.

   **Advantage:** You only need to perform the capture with the support page.

   **Disadvantage:** Users are not captured; you will have to set up all users you want to have access to the test environment.

If you need to bring claims from the production environment into a test environment for testing, the only way to do this is using the tool with database access to the production environment. These claims will contain PHI and all of the handling requirements of PHI will apply to both the extract that is created and the test system they are loaded in.

You can export claims by themselves without configuration as a separate step, so populating with claims can be done after the configuration is loaded. The saver.bat must be used to do the extract and CLAIMS must be selected. Typically, you will want to constrain the amount of claims you are extracting by a date range. (If you want to make a full mirror of the production environment with all claims, a database backup and restore would probably be a better option.)
2.6.2 Updating a production environment from a test environment

The loader and extractor application can be used to update a production environment from a test or stage environment. If you have done this operation manually in the past, this will likely prove to be a great time saver. It is important to understand the operation that is taking place (updates are being made to the production database) and to follow best practices and safety guidelines.

Some operations will choose a three-tiered approach while others may prefer a simpler two-tiered approach. Three-tiered systems usually involve 1) a test system where rules are developed and configurations are manipulated; 2) a stage system where those configurations and rules can be run against claims and be properly vetted; and 3) the production environment where business operations will take place.

A two-tiered approach, by contrast, is simply #1 and #2 above.

2.6.3 Best Practices for updating production environments

1. It is good practice to use the regular load tools within the application to update the KB, LCD and PQRS data on both the test and production systems. Updating the test environment first with a new KB for verification should still continue.

These data loads updates should be done prior to loading a configuration because it provides an opportunity for the configuration loader's safety checks to alert about any unusual situation.

Note that updating the test environment as new KnowledgeBases are released for verification prior to loading in production should still continue.

If the KB, LCD and PQRS data loads are performed on the production environments prior to any configuration update from test, it ensures configuration can successfully be migrated with just the default settings. (RULES AND CONFIGURATION and OVERRIDES in the saver tool or the save in the URL/support page.)

Note: Following these practices will result in the fastest loads with the most safety checks and the least amount of downtime.

You will typically not want to move claims into production from anywhere else. The production environments should be the “source of truth” regarding claims.

Considering that the configuration loader will make changes to a production system, it is strongly recommended that you perform a backup on production using the saver before loading a configuration into production. If recommendations #1 and #2 above are followed, this backup can be done using the default options of RULES AND CONFIGURATION and OVERRIDES. This backup will provide other benefits beyond being able to restore to a previous configuration if the new configuration and rules affect production negatively. There will also be an inherent audit trail in the form of snapshots of the configuration at different times; since the data doesn't contain PHI, it can be archived more conveniently.

2.7 Claim Migration Tool

The Configuration Extractor Loader tool provides access to a Claim Migration tool which includes the ability to migrate a copy of all claims from one source database to another target database. The database types of the source and destination databases do not need to be the
same, so it is commonly used to migrate data from an Oracle database to an MS SQL Server
database or vice versa. (See Configuration Saver and Loader Tools for instructions to access
the Configuration Extractor Loader tool).

2.7.1 Recommendations and Requirements
1. A network connection to the source and target database is required. (Special software such
as the Oracle client is not required.)
2. It is recommended that the same product version be installed on both source and target
machines. (The tool will prompt you if the versions are different.)
3. The enterprises need to be the same between the source and target database for the claims
to copy. (This will naturally be the case if the loader.bat tool was used first to copy the
configuration of the source system to the target system. This is the normal scenario for the
use of this tool.)

2.7.2 Running the Claim Migration Tool
To run the Claim Migration tool,
navigate to the directory where the
Configuration Extractor Loader tool
was installed using Windows Explorer
and then double-click the
claimcopy.bat batch command file.
The ICP Claim Migration Utility
window will appear.

The Source Database Connection
Configuration tab appears. The
Source Database is the database that
includes the claims that you want to
copy.

**CONNECTION CONFIGURATION**

1. Select your Database Type for the Source database (Oracle or MSSQL). The Database
URL should appear by default as well as the User (icp_p) and an empty Password field.
Note that the Database URL field will change depending on the type of database that was
selected. If the database server for the CM/CES system whose claim data is to be captured
is not the local machine, change “localhost” to the DNS hostname of the remote database
server.

<table>
<thead>
<tr>
<th>Note</th>
<th>Remote Desktop access to either the CM/CES server or the database server is not required. The tool only requires that the database be accessible on the network from which it is run and a valid account for connecting to the database.</th>
</tr>
</thead>
</table>

2. Click the Test Connection button to verify that your source database connection settings
are valid. If everything is correct, a success dialog box will pop up. If not, an error dialog is
displayed.
3. Click the **Target Database Connection Configuration** section and fill in the required connection configuration details for the target database where you would like the claims to be copied.

![Target Database Configuration](image)

4. Click the **Test Connection** button to verify that your target database connection settings are valid.

5. Click the **Copy** button and a “Ready to copy” message will appear with a notification that any existing claims on the target machine will be deleted.

6. Click the **Continue** button to proceed.

<table>
<thead>
<tr>
<th>Note</th>
<th>Prior to the copy, safety checks are performed by comparing the data from the source system to the target system. These safety checks allow you to cancel the load and leave the target machine unaltered.</th>
</tr>
</thead>
</table>

**Safety Check Examples:**

- **Different product versions between source and target machine:** If there is a table column mismatch between the source and target system (which usually happens when the databases have different product versions), a warning message will appear. The specific column details will be described and at this point you can choose to continue or cancel the migration.

- **Different enterprises between source and target machine:** If the enterprises are not consistent between the source and target machine it will notify you that it is not a match for claim copy.
7. If there are no safety concerns, a **Load Progress** dialog appears and shows each step of the claim copy process as it occurs.

8. When the claim copy process completes, the **OK** button is activated. Click the **OK** button to be returned to the main screen.

Claims are now successfully copied to the target database, so you can exit the Claim Migration tool.
3 Performance Monitor

3.1 Introduction
The Performance Monitor is a standalone system monitoring application for use in high-availability Claims Manager (CM) and Claim Edit System (CES) environments. The Performance Monitor can remotely connect to one or multiple CM/CES processing nodes and dynamically return performance metrics across a cluster in real time.

3.2 Prerequisites
CM/CES 5.4 Service Pack 1 (with High Availability enabled for clustered environments).

3.3 Installation and Quick Start
1. Download the PerformanceMonitor.zip installation package from the Optum Client Portal.
   Extract the file contents on to a server with network access to the CM/CES environment. Optum recommends installing on a separate server so that the Performance Monitor and CM/CES are not competing for hardware resources.
2. Open the Performance Monitor directory when the zip file extraction is complete.
3. Double-click the monitor.bat file to launch the Performance Monitor.
4. Enter the **IP address** or server name of the primary CM/CES node in the Server field.
   At least one **Server** must be specified. Secondary CM/CES nodes may also be specified in the Server field (comma separated), but is not required.

5. Add the **port number** to each IP address or server name listed in the Server field.
   The Performance Monitor connects to CM/CES using port 61616. See the following figure.

6. Click **Connect** to open a connection to the target server.
Upon successful connection, all available processing and connector nodes within the CM/CES cluster are displayed (as shown above). The Performance Monitor automatically begins real-time monitoring and performance metrics for each node will dynamically update in real-time.

3.4 SSL Setup

The following instructions describe how to secure communications between the Performance Monitor and CM/CES using SSL.

**Note** Ensure SSL is already set up and enabled within the CM/CES cluster before attempting to securely connect via the Performance Monitor.

1. Navigate to the `Optum/ICP/conf` directory on the primary CM/CES connector node.

2. Copy the truststore file *(truststore.ks)*. The truststore file contains all of the imported public certificates for all the CM/CES nodes participating in the cluster.

3. Paste the truststore file in the root Performance Monitor directory, as shown at right.
The Performance Monitor truststore must minimally contain the public certificates for the main and standby CM/CES connector nodes. Copying this file from the primary connector node—which already holds the necessary public certificates—is one way to quickly secure connections from the Performance Monitor. If SSL is enabled on the CM/CES connector, this truststore file may also contain various certificates from other client servers sending claims into the CM/CES connector.

4. Double-click the monitor.bat file to launch the Performance Monitor.
5. Enter the IP address or server name of the primary CM/CES node in the Server field.
6. Add the port number to each IP address or server name listed in the Server field.

SSL connections use port 61617. See the following figure.

7. Check the Use SSL checkbox.
8. Click Connect to open a secure connection to the target server.

Upon successful connection, all available processing and connector nodes within the CM/CES cluster are displayed (as shown above). The Performance Monitor automatically begins real-time monitoring. Performance metrics for each node will dynamically update in real time.

The System tab is divided into three main sections:

- CPM/LPM Metrics
- Connectors table
- Recent Interesting Claims
3.5 CPM/LPM Metrics

The Performance Monitor reports real-time Claims per Minute (CPM) and Lines per Minute (LPM) rates for each CM/CES processing node over various time intervals.

The Performance Monitor displays the following data for each processing node:

- **Status**: Indicates whether the CM/CES processing node is available or not.
- **Node**: The server name or IP address of the CM/CES processing node.
- **CPM 1**: Claims/minute rate measured over the last minute.
- **LPM 1**: Claim lines/minute rate measured over the last minute.
- **CPM 5**: Claims/minute rate measured over the last five minutes.
- **LPM 5**: Claim lines/minute rate measured over the last five minutes.
- **CPM 10**: Claims/minute rate measured over the last ten minutes.
- **LPM 10**: Claim lines/minute rate measured over the last ten minutes.
- **CPM 20**: Claims/minute rate measured over the last 20 minutes.
- **LPM 20**: Claim lines/minute rate measured over the last 20 minutes.
- **Total Claims**: Total number of claims processed in the current session (i.e., based on when the Performance Monitor was started).
- **Total Lines**: Total number of claim lines processed in the current session.
- **Total History**: Total number of history claims received in the current session.

CPM and LPM rates update dynamically as claims are received and processed within CM/CES.

3.6 Connectors Table

The Connectors table identifies servers within the CM/CES environment where ICP Connector services are running. Connector nodes are responsible for receiving incoming claims and brokering requests to the processing nodes. Consequently, at least one connector node—a primary connector—must be available in order to process claims. A second connector node—or standby connector—can be configured in CM/CES when High-Availability (HA) is enabled. The Performance Monitor displays both the main and standby connectors if both are available.

All requests are brokered through the primary connector, by default. However, CM/CES will automatically failover to a standby connector in the event the primary connector shuts down or becomes unresponsive.
The Connectors table displays the following for each connector node:

- **Node**: The server name or IP address of the connector node.
- **Claims Received**: Total number of incoming claim requests.
- **Claims Queued**: Total number of requests sent to CM/CES for processing.
- **Results Received**: Total number of requests receiving an editing response from CM/CES.
- **Results Undeliverable**: Total number of responses that failed to return to the connector node (e.g., as a result of a timeout).

### 3.7 Recent Interesting Claims

The Recent Interesting Claims table identifies, in real time, any individual claims that encounter either of the following conditions during CM/CES processing:

- An **ERROR-level log statement occurs anywhere in the CM/CES claim flow**
- Claim-processing time exceeds the Performance Monitor’s user-defined threshold (i.e., a long-running claim)

The threshold for claim-processing time defaults to 30 seconds; however, you can select from the **Acquire detailed metrics for processing times longer than** dropdown to lower or raise this setting.

The Recent Interesting Claims table displays the following summary information for each reported claim:

- **Reason**: Describes why the claim was reported (e.g., “Failed Claim Analysis”).
- **Node**: The server name or IP address of the CM/CES processing node where the claim was processed.
- **Time**: Date and timestamp when the ERROR occurred or processing-time was exceeded.
- **Duration**: Claim processing time (in milliseconds).
- **Claim EID**: The external Claim ID cited on the claim.
- **Route**: Name of the Claim Route the claim was directed to.
- **Thread**: Name of the thread in CM/CES that the associated claim was running on. The thread name also appears in all logs associated with the claim processing.

**Note**

“Interesting” claims are saved to a local database file in the folder where the performance monitor has been installed (e.g. `C:\PerformanceMonitor`). The file name is of the format: `metrics_YYYYMMDD_HHmmSS.mv.db` so as to be named with the date and time that the file is created. These files are extremely important for debugging performance problems as the data they contain can identify for each “interesting” claim exactly which part of the claim analysis is consuming the bulk of the time. Using this file is discussed in detail in the section.

### 3.8 View Claim Details

Follow the instructions below to view additional claim-flow or error information for claims reported in the Recent Interesting Claims table:

1. Double-click any row within the Recent Interesting Claims table. Claim details display in a separate window.
The Claim Flow tab displays by default. The Claim Flow identifies when notable events in the claim processing workflow occurred and how much time those events required for the selected claim.

2. Click the ERROR tab to view errors reported for the selected claim.

The Error tab identifies errors reported in the CM/CES log files for the selected claim.

3. Close the claim details window to return to the Performance Monitor.

3.8.1 Performance View

The Performance tab reports real-time performance metrics across all nodes within CM/CES environment in the form of area and line graphs. The Performance tab provides three types of reports:

- Cluster Performance Graph
- Node Comparison Graph
- Average Throughput Time

3.8.2 Cluster Performance Graph

The Cluster Performance Graph dynamically plots an area graph that illustrates the aggregate performance of a CM/CES cluster and the contribution of individual processing nodes. Performance increases up the vertical axis. Time is represented along the horizontal axis. Data points are plotted based on calculated node performance over a user-defined time interval, or Sample Rate.
The Cluster Performance Graphs provides the following user settings:

- **Sample Rate**: Determines how frequently the Performance Monitor calculates performance, i.e., how often new data points are plotted. By default, the Sample Rate is set to five (5) seconds.
- **Use lines as metric**: Changes the performance metric from claims to claim lines.
- **Raw Count**: Measures performance based on the total number of claims or lines processed within a time interval (based on the Sample Rate)—as opposed to a calculated CPM or LPM value.

### 3.8.3 Node Comparison Graph

The Node Comparison Graph dynamically plots line charts that illustrate the individual performance of CM/CES processing nodes relative to other nodes in a cluster. Performance increases up the vertical axis. Time is represented along the horizontal axis. Data points are plotted based on calculated node performance over the **Sample Rate** interval.
Like the Cluster Performance Graph, the Node Comparison Graph also provides settings for users to control **Sample Rate, Raw Count, and Use lines as metrics.**

### 3.8.4 Average Throughput Time

The Average Throughput Time Graph dynamically plots an area graph that illustrates total average processing time per claim (in milliseconds). The graph also illustrates how the following steps in the claim processing workflow affect overall throughput:

- **Translation**: Time required to translate the external claim data format and pass it to the CM/CES queue.
- **In queue**: Amount of time a claim is held in queue before being passed to the CM/CES rules engine.
- **Rules**: Amount of time CM/CES requires to route a claim to an appropriate ruleset and conduct clinical editing.
- **Out queue time**: Amount of time an edited claim is held in queue before being returned to the client interface.

Average processing time (in milliseconds) increases up the vertical axis. Time is represented along the horizontal axis. Data points are plotted according to **the Sample Rate** interval.
The **Raw Count** and **Use lines as metrics** settings are not supported when viewing the Average Throughput Time graph.
3.8.5 Advanced View

The Advanced tab provides access to the Performance Monitor’s in-memory database. This provides system and/or database administrators with the flexibility to query the underlying performance data in specific ways or dimensions.

The Performance Monitor uses volatile data storage—meaning the data is persisted and available within the database only while the Performance Monitor is running. Upon shutdown, all data is cleared from the database and archived in an external database file (.mv) within the Performance Monitor root directory (as shown in the following figure).
The database files may be used by Optum support representatives to investigate and troubleshoot any performance issues that may arise.
1 Report Queries

This section outlines how to use queries in a SQL query editor to produce data the following reports:

- Rules in Ruleset for ILOG (including flag status)
- Rules in Ruleset for DDR (including flag status)

1.1 How to Run the Report Queries

1. Connect to the database using a SQL query editor, such as SQLplus, Toad for Oracle, or SQL Server Management Studio for Microsoft SQL Server. Log in as the ICP_P user.

2. Copy the query for the desired report from one of the sections below, then paste it into the SQL editor screen. (Alternatively, you may paste the query into a text editor and save it as a file on a local hard drive (e.g. c:\temp\myquery.sql)

3. Execute the query by pressing the execute button in the SQL query editor.

Notes

- For command-line based query editors, such as SQLplus, it is more practical to execute a query that has been saved to a file, (e.g. SQL>@c:\temp\myquery.sql)
- If the query will not run correctly, ensure the quotes are in plain text format instead of “fancy quotes” that are typically used by a word processor. (e.g., “instead of “).
- Optum may provide queries to clients for use in specific circumstances. If the client chooses to use such queries elsewhere or in another manner, the client is solely responsible for the use of such queries, including any errors, omissions or other faults.

1.2 Rule in Ruleset Reports

Since Data-Driven Rules are fundamentally different from ILOG rules, separate queries are required for each rule type. Ruleset reports only apply to static rulesets. Corresponding reports for DDR usage in the dynamic rulesets are under development.

1.2.1 ILOG Rules in Ruleset Query (Includes flag status)

Purpose: This section contains separate queries for Oracle and MS SQL Server. The data returned has the same fields as the rules in ruleset(s) report in the product. It also includes the flag status within each ruleset. The following fields are returned by the query:

- Name
- Version
- Rule Status
- Rule Set
- Mnemonic
- Flag Status
- Environment
- Priority
- Ruleset Enterprise Name
- Rule Enterprise Name

Sample query results: In the sample below, the data output from the query has been placed into Excel so the data can be sorted.
Product Versions: 5.2.1 +

Prerequisites: None

Oracle Query — ILOG Rules in Ruleset:

MS SQL Server Query — ILOG Rules in Ruleset:

1.2.2  DDR Rules in Static Ruleset Query (Includes flag status)

Purpose: This section contains separate queries for Oracle and MS SQL Server. The data returned has the same fields as the rules in the rulesets report in the product but also includes the flag status within each ruleset. The following fields are returned by the query:

Name|Version|Rule Status|Rule Set|Mnemonic|Flag Status|Environment|Priority|Ruleset Enterprise Name|Rule Enterprise Name

RULESET_NAME|PATTERN_ID|PATTERN_NAME|ENTERPRISE_NAME|PARENT_ENTERPRISE|LINE_OF_BUSINESS|IGNORE_LOB|ENABLED|EFFECTIVE_DATE|EXPIRATION_DATE|RUN_ORDER|MNEMONIC|FLAG_STATUS

Sample query results: In the sample below, the data output from the query has been placed into Excel so the data can be sorted.

Product Versions: 5.2.1 +

Prerequisites: None

Oracle Query — DDR Rules in Ruleset:

MS SQL Server Query — DDR Rules in Ruleset:
2  Delete Inactive LCDs in Bulk

**Purpose:** When a user attempts to remove a large amount of inactive LCD data, the UI processes for a few minutes and eventually times out. If the user selects smaller amounts of inactive LCD data to remove, the UI works as expected. The following scripts allow the user to set a date and all Inactive LCD data will be removed prior to that date.

**Product Versions:** 5.2.1 +

**Prerequisites:** None

**Before Running the Query**

Certain “tags” must be replaced with meaningful values before executing these SQL scripts. The tags are enclosed in angle brackets and highlighted in RED for easy visibility.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;EXTRACTDATE&gt;</code></td>
<td>Change to the date value for which the Inactive LCD’s will be removed. This value corresponds to the LCD version date in the UI. (FORMAT = YYYYMMDD)</td>
</tr>
</tbody>
</table>

**Oracle Database Query – Delete Inactive LCDs in Bulk:**

BulkDelete_InactiveLCD
Ds_Oracle.txt

**MS SQL Server Database Query – Delete Inactive LCDs in Bulk:**

BulkDelete_InactiveLC
Ds_SQLServer.txt
3 Remove Calendar Month LCD Frequencies

**Purpose:** As of the 2020 Q1A KB, the DDR LCD rule supports a “Calendar Month” frequency option. In addition to the rule change, LCD data updates are required for the DDR rule to appropriately apply these frequency edits. When LCD data updates are provided on the customer portal that include the new “Calendar Month” option (PERIOD_UNIT set to 'F' in LCD_FREQUENCY.txt file), the ILOG LCD rule will not recognize this option during claim analysis and evaluation against those policies with Calendar Month frequencies will result in incorrect editing.

For clients that still use the ILOG LCD rule, a special database script is provided to reset the PERIOD_UNIT to a value that is recognized by the ILOG LCD rule so as to maintain backwards compatibility. The data for these frequencies is also moved from pay groups into profile groups. Execute the script below that is appropriate for your database type (Oracle or MS SQL Server).

| Note | ILOG LCD users will need to run this script after loading LCD updates (using the UI or the LCDBulkloader script) that include Calendar Month frequencies. This must be done after every LCD update once the data changes have been made. Optum will notify clients once this has been completed; currently targeted for Q1 2020. DDR LCD users do not need to run this script; therefore, migrating off of the ILOG LCD rule to the new, more efficient DDR LCD rule is strongly recommended. |

**Product Versions:** 5.3.1 +

**Prerequisites:** The 2020 Q1A KB or later has been installed.

**Technical Details**

The script performs the following operations:

- Finds records that are in pay groups and contain Calendar Month frequencies (period unit of 'F' in LCD_FREQUENCY) and writes them to a temporary table.
- Writes ID mappings of the pay group and the new corresponding profile group to a temporary table.
- Creates new profile groups corresponding to the pay groups with Calendar Month frequencies so that a pay/profile pair of groups are generated.
- Add records for all attributes for the new profile group. (This assumes that all attribute records are identical to the attributes in the pay group.)
- Updates the frequency in the profile group from “Calendar Month” to “Month by Date”.
  - Change equality operator to 'GT' and adjust the period to the original calendar period minus 1. (Example: “12 Calendar Months” adjusted to 11 months by date).
- Updates the frequency from “Calendar Month” to “Month by Date”.
  - Adjust the period to the original calendar period minus 1. (Example: “12 Calendar Months” adjusted to 11 months by date.)
- Cleans up the temporary tables.
Running the Query

1. Load updated LCD data using the CM/CES user interface or with the LCDBulkLoader utility.
2. Using a query editor (SQL Developer, MS SQL Server Management Studio, etc.) connect to the ICP database as the "icp_p" user.
3. Copy the appropriate script for your database from the pages below into your query editor and then execute the script.
4. Restart the ICP services to force the database changes to be recognized by the software.

Oracle Database Query – Remove Calendar Month LCD Frequencies:

MS SQL Server Database Query – Remove Calendar Month LCD Frequencies:
4 Database User Permissions

ICP products (Claims Manager and Claims Edit System) rely on a relational database to store data (KnowledgeBase, application configuration, claims, etc.). These applications are designed to be self-managing/self-monitoring with regard to the database, because the majority of clients do not have DBAs on staff to manage these tasks.

When running in production, the application accesses the database with a single “Product User” account. However, other user accounts with elevated privileges are required when performing the initial product installation and when updates are applied. At the time of the initial installation, a user accounts with DBA privileges (i.e., SYSTEM or SA) creates the database and the product user accounts (ICP_IS and ICP_P), and performs the initial database configuration for the application. Installation of a Cumulative Update may also alter the database schema in various ways that require elevated privileges.

In total, the following four accounts are used by an ICP product. The purpose for each of these accounts and the functions they perform are described in the table below.

<table>
<thead>
<tr>
<th>User Type</th>
<th>User Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System DBA</td>
<td>SYS as SYSDBA (Oracle only)</td>
<td>This is an account with database administration privileges. It is used by the ICP Service (Bifrost) to grant specific permissions for JBoss XA transaction management that can only be granted by this user. (See Note in section 0 below.)</td>
</tr>
<tr>
<td>System Admin</td>
<td>SYSTEM (Oracle) SA (SQL Server)</td>
<td>This is an account with database administration privileges. It is used by the ICP Service (Bifrost) to create the ICP_IS user and grant it the necessary privileges to create the database and other administrative tasks.</td>
</tr>
<tr>
<td>Product Admin</td>
<td>ICP_IS</td>
<td>This user account is the application database admin account. It is only used by the ICP Service (Bifrost) which is the component responsible for running database schema migration scripts when the product software is installed or updated. This account is used to create the ICP_P user and the underlying storage structure. For Oracle, this includes creating the tablespaces that are required by the application (DATA1, INDEX1, and FREYA). After the database structure has been created, Bifrost from then on only verifies its existence as a monitoring function, but does not alter the database structure and therefore no longer requires DBA privileges.</td>
</tr>
<tr>
<td>Product User</td>
<td>ICP_P</td>
<td>Users of ICP products do not directly login to their database. They login to the application and the application connects to the database as the Product User (ICP_P). This user account is the application schema owner. It is used by the application for all connections to the database for claim processing and data updates via the UI including loading of data via CSV files.</td>
</tr>
</tbody>
</table>
Typical application schema privileges are assigned to this account plus one or more system privileges or roles depending on the database type. These are required by the “Claim Purge” function to schedule database jobs and for Managing XA transactions.

**Oracle:** CREATE JOB. (See below for details on privileges for managing XA transactions.)

**SQL Server:** SQLAgentUserRole, SQLAgentReaderRole, SQLAgentOperatorRole on the msdb database and SQLJDBCXAUser on the master database.

Also note that because the application is multi-threaded, there will be multiple connections to the database for each claim processing thread. In addition, there will be a connection for each user that is logged into the application. Connections are obtained from an application-side connection pool for maximum performance.

### Note

Passwords for the above accounts are stored in an encrypted form.

- The `services.properties` file which is used by Bifrost stores passwords for the Product User, Product Admin, and System Admin user accounts.
- The `standalone.xml` file that is used to configure JBoss (Wildfly) stores the password for the Product User only.
- The password for the System DBA account is only used by the installer and is not stored.

## 4.1 Product Admin and Product User Permissions (Oracle)

The tables below identify the privileges that are granted to the Product Admin (ICP_IS) and Product User (ICP_P) accounts. The purpose for each privilege is described.

<table>
<thead>
<tr>
<th>Product Admin Privileges</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GRANT CREATE SESSION TO ICP_IS;</strong></td>
<td>Allows initial connection to the database.</td>
</tr>
</tbody>
</table>
| **GRANT DBA TO ICP_IS;** | - Initial tablespace creation and management of Bifrost. Bifrost verifies tablespaces, data files, and log files.  
- Granting the CREATE JOB privilege to the ICP_P user which is required in order to purge claims.  
- Perform database updates when an ICP software update is applied. This ensures that the database schema is compatible with the new ICP software release. This frequently requires schema changes. |
### Product Admin Privileges

<table>
<thead>
<tr>
<th>Grant / Permission</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANT CREATE USER TO ICP_IS; GRANT DROP USER TO ICP_IS;</td>
<td>Allows the ICP_IS user to create the ICP_P user. The grant to drop users is used by Development.</td>
</tr>
<tr>
<td>GRANT CREATE SEQUENCE TO ICP_IS; GRANT CREATE TABLE TO ICP_IS; GRANT CREATE TRIGGER TO ICP_IS; GRANT CREATE VIEW TO ICP_IS;</td>
<td>Allows the ICP_IS user to create database tables and other schema objects that are owned by the user.</td>
</tr>
<tr>
<td>GRANT UNLIMITED TABLESPACE TO ICP_IS;</td>
<td>The ICP_IS user actually only owns one small table (COMPONENT_VERSIONS), but this privilege is added for consistency with the ICP_P user.</td>
</tr>
</tbody>
</table>

### Product User Privileges

<table>
<thead>
<tr>
<th>Grant / Privilege</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANT CREATE SESSION TO ICP_P; GRANT CONNECT TO ICP_P; GRANT RESOURCE TO ICP_P; GRANT ALTER SESSION TO ICP_P;</td>
<td>Allows the ICP_P user to connect to the database and configure session parameters.</td>
</tr>
<tr>
<td>GRANT CREATE SEQUENCE TO ICP_P; GRANT CREATE TABLE TO ICP_P; GRANT CREATE TRIGGER TO ICP_P; GRANT CREATE VIEW TO ICP_P; GRANT CREATE ANY SYNONYM TO ICP_P; GRANT DROP ANY SYNONYM TO ICP_P; GRANT CREATE PROCEDURE TO ICP_P;</td>
<td>Allows the ICP_P user to create database tables and other schema objects that are owned by the user.</td>
</tr>
<tr>
<td>GRANT CREATE JOB TO ICP_P;</td>
<td>Allows the ICP_P user to create database jobs. This is required for scheduling claim purge jobs and scheduled reports which execute stored procedures. Privileges granted to a user in a role are not available in this case, so this grant must be given directly to the user rather than being assigned via a role.</td>
</tr>
<tr>
<td>GRANT UNLIMITED TABLESPACE TO ICP_IS;</td>
<td>The application uses three tablespaces exclusively (DATA1, INDX1 and F_REYA). We grant UNLIMITED because many of our customers do not have a DBA to monitor the space usage. However, if available, quotas could be set and maintained by the DBA staff.</td>
</tr>
</tbody>
</table>
### Product User Privileges

<table>
<thead>
<tr>
<th>Grant / Privilege</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANT SELECT ON SYS.DBA_PENDING_TRANSACTIONS TO ICP_P;</td>
<td>These are the required permissions for the ICP_P database user to use XA Transactions. <strong>Note:</strong> These grants are explicit user grants, meaning you must grant them to a user while logged in as SYSDBA (typically as the SYS user). This is in accordance with Oracle’s security policy.</td>
</tr>
<tr>
<td>GRANT SELECT ON SYS.PENDING_TRANS$ TO ICP_P;</td>
<td></td>
</tr>
<tr>
<td>GRANT EXECUTE ON SYS.DBMS_XA TO ICP_P;</td>
<td></td>
</tr>
<tr>
<td>GRANT SELECT ON SYS.DBA_2PC_PENDING TO ICP_P;</td>
<td></td>
</tr>
<tr>
<td>SYS.DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE (PRIVILEGE =&gt; 'DEQUEUE_ANY')</td>
<td>The application uses Oracle’s <em>Advanced Queueing</em> feature. These grants are required to maintain and use the queuing functionality. This is only used by Statistical Editing that is a part of the ACE functionality (enabled via a special license).</td>
</tr>
<tr>
<td>SYS.DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE (PRIVILEGE =&gt; 'ENQUEUE_ANY');</td>
<td></td>
</tr>
<tr>
<td>GRANT EXECUTE ON SYS.DBMS_AQ TO ICP_P;</td>
<td></td>
</tr>
<tr>
<td>GRANT EXECUTE ON SYS.DBMS_AQADM TO ICP_P;</td>
<td></td>
</tr>
<tr>
<td>GRANT EXECUTE ON SYS.DBMS_AQ_BQVIEW TO ICP_P;</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.1.1 Managing Privileges for the Product Admin User (Oracle)

In many enterprises, the privileges granted to users is strictly monitored. It is not uncommon for concerns to be raised with regard to the database administrative privileges that are granted to the Product Admin user (ICP_IS). Because this user is only needed during the installation and upgrade of the software, there is no problem if the high-level privileges are revoked after the installation or upgrade has completed. However, before performing the next software update, they must be granted again.

The following SQL scripts can be used to revoke and grant database administrative privileged for the Product Admin user (ICP_IS). Note that they are delivered with the product and can be found in the `<InstallDir>/bin` directory.

**revoke_dba-icp_is.sql**

```sql
REVOKE dba FROM icp_is;
GRANT select on dba_data_files TO icp_is;
GRANT select on dba_tablespaces TO icp_is;
```

**grant_dba-icp_is.sqlb**

```sql
GRANT dba TO icp_is;
```
### 4.2 Product Admin and Product User Permissions (SQL Server)

The tables below identify the privileges that are granted to the Product Admin (ICP_IS) and Product User (ICP_P) accounts. The purpose for each privilege is described.

These privileges can be viewed and modified using SQL Server Management Studio by navigating to the Security → Logins folder, then by double-clicking either the icp_is or icp_p login account. The Login Properties dialog is then displayed with pages for Server Roles and User Mapping which is where the following roles can be viewed.

<table>
<thead>
<tr>
<th>Product Admin Privileges</th>
<th>(ICP_IS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Role Membership</strong></td>
<td><strong>Server / DB</strong></td>
</tr>
<tr>
<td>bulkadmin</td>
<td>Server Role</td>
</tr>
<tr>
<td>public</td>
<td></td>
</tr>
<tr>
<td>sysadmin</td>
<td>Server Role</td>
</tr>
<tr>
<td>securityadmin</td>
<td></td>
</tr>
<tr>
<td>public</td>
<td>master</td>
</tr>
<tr>
<td>db owner</td>
<td>icp</td>
</tr>
<tr>
<td>db accessadmin</td>
<td>icp</td>
</tr>
<tr>
<td>db ddladmin</td>
<td>icp</td>
</tr>
<tr>
<td>public</td>
<td></td>
</tr>
<tr>
<td>db securityadmin</td>
<td>icp</td>
</tr>
<tr>
<td>db datareader</td>
<td>icp</td>
</tr>
<tr>
<td>db datawriter</td>
<td>icp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product Admin Privileges</th>
<th>(ICP_P)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Role Membership</strong></td>
<td><strong>Server / DB</strong></td>
</tr>
<tr>
<td>bulkadmin</td>
<td>Server Role</td>
</tr>
<tr>
<td>public</td>
<td></td>
</tr>
<tr>
<td>Role Membership</td>
<td>Server / DB</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>public SQLAgentOperatorRole</td>
<td>msdb</td>
</tr>
<tr>
<td>SQLAgentUserRole</td>
<td></td>
</tr>
<tr>
<td>SQLAgentReaderRole</td>
<td></td>
</tr>
<tr>
<td>db owner</td>
<td>master</td>
</tr>
<tr>
<td>public</td>
<td>master</td>
</tr>
<tr>
<td>SqlJDBCXAUser</td>
<td>master</td>
</tr>
<tr>
<td>ddladmin</td>
<td>icp</td>
</tr>
<tr>
<td>public</td>
<td>icp</td>
</tr>
<tr>
<td>db datareader</td>
<td>icp</td>
</tr>
<tr>
<td>db datawriter</td>
<td></td>
</tr>
<tr>
<td>ingenix proc exec</td>
<td>icp</td>
</tr>
</tbody>
</table>
Glossary

A

Artemis
A third-party JMS provider that has been integrated into the JBoss application server, but that CM/CES uses external to JBoss, and is sometimes referred to as “the Broker” in CM/CES. JMS stands for the “Java Messaging Service” which is the middleware that allows application components based on the Java Enterprise Edition (Java EE) to create, send, receive, and read messages. In the case of CM/CES, it allows the Connector to exchange claim data with the core software running in the JBoss application server reliably and asynchronously.

B

BRES
Business Rules Execution System. The ILOG component that executes the claim analysis rules contained in a ruleset.

C

Certificate Authority
An entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate (i.e., a specific computer host name). This allows others (relying parties) to rely upon signatures or on assertions made about the private key that corresponds to the certified public key. In this model of trust relationships, a CA is a trusted third party—trusted both by the subject (owner) of the certificate and by the party relying upon the certificate. Many public-key infrastructure (PKI) schemes feature CAs.

CM/CES
Claims Manager/Claims Edit System. The software system that analyzes claims for clinical accuracy and overpayment. Historically, two product types have been distributed to support “providers” (Claims Manager) and “payers” (Claims Edit System), but these two product types will eventually be merged.

CSR
Certificate Signing Request. A block of encrypted text that is generated on the server that the certificate will be used on. It contains information that will be included in your certificate such as your organization name, common name (domain name), locality and country. It also contains the public key that will be included in your certificate. A private key is usually created at the same time that you create the CSR.

D

Digital Certificate
An electronic document used to prove ownership of a public key. The certificate includes information about the key, information about its owner’s identity, and the digital signature
of an entity that has verified the certificate’s contents are correct. If the signature is valid, and the person examining the certificate trusts the signer, then they know they can use that key to communicate with its owner. Ultimately, the certificate is used to generate a cryptographic key that is used to seed an encoding scheme that is unique to that connection, thus enabling secure encrypted communication.

H

HA

High Availability. This term refers to a system architecture where there are multiple copies of the main components so that if one component fails, the other identical copies can take over the full load so that the overall system remains “available” to external users. The goal is to eliminate single points of failure that will stop the overall system from functioning. CM/CES software accomplishes this by using multiple computers (nodes) as a single system, each of which can run all of the ICP services (ICP Service, ICP Engine, and ICP Connector) so that if one node fails, components on another node can take over. HA architecture also provides scalability to increase the overall performance of the system.

HTTP

Hypertext Transfer Protocol. An application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

HTTPS

HTTP Secure. A protocol for secure communication over a computer network which is widely used on the Internet. HTTPS consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer (SSL). The main motivation for HTTPS is authentication of the visited website and protection of the privacy and integrity of the exchanged data.

I

ILOG

Third-party rules engine software (the vendor is IBM) that is used to analyze claims for clinical accuracy.

J

Java EE

Java Enterprise Edition. The specification for application servers that provide middleware required to make enterprise applications that are modular, robust and portable to various computer hardware and operating system.

JBoss

The open source J2EE Application Server that is used to host the core components of the CM/CES software system.
JMS
Java Messaging System. A component of the J2EE specification that defines how messages are delivered over the network. A JMS “provider” is the actual software that provides this service.

JTA
J2EE Transaction Architecture. The part of the J2EE specification that defines how multiple resources (multiple databases, messaging systems, etc.) are combined into a single transaction using a two-phased commit. Only after all resources involved in the transaction have reported their ability to successfully complete the transaction is the overall transaction committed.

JVM
Java Virtual Machine. The runtime environment for Java software that interprets the Java byte-code.

LDAP
The Lightweight Directory Access Protocol is commonly used for communicating with a central repository of usernames and passwords (a directory of user names). Microsoft’s “Active Directory” is one of the more popular of such products.

O
OCA
Optum Claims Accuracy. A temporary name change for the CM/CES software. Only the 5.2.1 release employed this name and used “OCA” as the install folder instead of “CM/CES”. When working on an installation 5.2.1 software, please substitute “OCA” for “CM/CES” in path names specified in this document.

P
PCS (Patient Claim Serialization)
Patient Claim Serialization is managed by a module in the Claim Connector. This module tracks each incoming claim via a separate queue for each unique patient MRN. During the time that a claim is being analyzed, any other claims that are received and which are for the same patient as the claim being analyzed are held in the queue for that patient until the claim response is received. This event releases the next waiting claim for that patient. When the claim response for the last claim that is queued for that patient is received, the queue is deleted

Public Key Certificate
Alternate name for Identity Certificate or Digital Certificate (see Digital Certificate).

R
Root Certificate
In cryptography and computer security, a root certificate is an unsigned or a self-signed digital certificate that identifies the root certificate authority (CA). A root certificate is part of a public key infrastructure scheme. The most common commercial variety is based on the ITU-T X.509 standard, which normally includes a digital signature from a certificate authority (CA).
Digital certificates are verified using a chain of trust. The trust anchor for the digital certificate is the Root Certificate Authority (CA).

**SSL**

Secure Sockets Layer. A cryptographic protocol that provides communication security over a computer network. Several versions are in widespread use in applications such as web browsing, email, Internet faxing, instant messaging, and voice-over-IP (VoIP). Major web sites use TLS (the latest version of the protocol) to secure all communications between their servers and web browsers. The term, SSL, was coined by the inventors of the first versions of the protocol, Netscape (the company was later bought by AOL).

**Transport Layer Security (TLS)**

TLS is the new name for SSL. Namely, SSL protocol got to version 3.0; TLS 1.0 is "SSL 3.1". TLS versions currently defined include TLS 1.1 and 1.2. Each new version adds a few features and modifies some internal details. We sometimes say "SSL/TLS".

The name was changed from SSL to TLS to avoid any legal issues with Netscape so that the protocol could be "open and free" (and published as an RFC). It also hints at the idea that the protocol works over any bidirectional stream of bytes, not just Internet-based sockets.

**URL**

Uniform Resource Locator. Commonly termed a web address, it is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier (URI). URLs occur most commonly to reference web pages (http), but are also used for file transfer (ftp), email (mailto), database access (JDBC), and many other applications.
Document Revision History

The following table outlines the revision history for this document.

- The most recent changes are added to the top of the table and oldest at the bottom.
- The revision numbering is for the document as a whole, not for the specific chapter to which it belongs. Therefore, when a change is made anywhere in the document, the overall document revision number must be incremented.
- The CH column refers to the chapter in which the revision was made.
  - IN = Introduction
  - TC = Technical Configuration
  - CT = Command-Line Tools
  - ST = Support Tools
  - QT = SQL Tools

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>CH</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14-May-2020</td>
<td>1.28</td>
<td>TC</td>
<td>• Added more detail to the <strong>Prerequisites</strong> note in section 3.5.2 - Adding a Connection to a Reporting Database. More requirements are specified for how the replication must be configured and what data tables and stored procedures must be replicated.</td>
</tr>
<tr>
<td>13-Mar-2020</td>
<td>1.27</td>
<td>QT</td>
<td>• Replaced in-line text SQL scripts with embedded text files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the LCDCalendarMonthDataMigrationForILOG_Oracle SQL script to correct an error.</td>
</tr>
<tr>
<td>10-Jan-2020</td>
<td>1.27</td>
<td>QT</td>
<td>• Added section <strong>4 - Database User Permissions</strong>.</td>
</tr>
<tr>
<td>13-Dec-2019</td>
<td>1.27</td>
<td>QT</td>
<td>• Added section <strong>3 - Remove Calendar Month LCD Frequencies</strong>.</td>
</tr>
<tr>
<td>13-Dec-2019</td>
<td>1.27</td>
<td>ST</td>
<td>• Minor corrections to section 2.5.1 Capturing a Configuration (command-line) and section 2.5.2 Loading a Configuration (command-line).</td>
</tr>
<tr>
<td>8-Oct-2019</td>
<td>1.26</td>
<td>TC</td>
<td>• Added section <strong>3.15 Reduce Claim Purge DB Blocking</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the note in section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added information to section 5.5.1 Configuring the Standalone.xml File with regard to setting the password for the keystore and truststore files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added section <strong>5.5.2.1 Updating the Keystore Password in Batch Scripts</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added section <strong>5.5.4.1 Configuring the connector.wrapper.conf File</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added section <strong>3.7 Changing the Keystore/Truststore Passwords for HTTPS</strong>.</td>
</tr>
<tr>
<td>8-Oct-2019</td>
<td>1.26</td>
<td>CT</td>
<td>• Update to <strong>3.10 - Claim Line Expansion</strong> section to include a new configuration parameter added to control the maximum number of units for which the “Expand Lines” function will apply.</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>CH</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>----</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 03-May-2019 | 1.25    | ST | • Update to Support Tools chapter [2.4.2 - Capturing a Configuration](#) to section 5 for Saver tool:  
  o Added information about what is included when the saver.bat Reports option is selected.  
  o Removed the option for User Defined Tables.  
  • Created a new section in the Support Tools chapter for [2.7 - Claim Migration Tool](#) which includes the ability to copy all claims from a source database to a target database.  
  • Update to section Cross Claim Type Editing to indicate that two separate configuration properties are required to enable this feature. Documented the SQL statements to add/update these properties.  
  • Added section [3.5.2 - Adding a Connection to a Reporting Database](#) to provide instructions for allowing reports to be generated using a reporting database that is replicated from the production database.  
  • Screenshots of the Loader, Saver, and Claim Migration tools no longer show the version number. The screen shots will only be updated if the content changes, not just the version number. |
| 15-Mar-2019 | 1.24    | ST | Update to Support Tools section [2.3 - URL/Support Page Method](#) which includes an updated screenshot displaying:  
  • Readability enhancements such as increased font and spacing between sections  
  • **Include specific de-identified claims** option no longer displays the text box for entering claim IDs unless the check box is checked.  
  o Once checked, the text box appears and includes a Claim IDs header.  
  o If checkbox for **Include history claims for any claims specified above** is selected, then the second checkbox option will appear stating **Save history claims in separate file.**  
  Other Support tool enhancements:  
  • Informational pop-up box widths have been reduced to make them easier to read  
  • PHI informational message has been moved above the download button |
| 15-Feb-2019 | 1.23    | TC | • Major revision to the structure of the document. Divided clearly into chapters. Section numbering starts over at 1 for each chapter.  
  o Introduction to This Guide  
  o Technical Configuration  
  o Command-Line Tools  
  o Support Tools  
  o SQL Tools  
  • Added section [5.5.3.2 - Configuring the icp.wrapper.conf File](#) which explains that when the keystore pass is changed, it must be updated for the ICP Service in addition to other places.  
  • Added more information to section [3.13.5 - Handling Large Claim Messages](#) to explain the connection between the minLargeMessageSize parameter in jndi.properties and the <journal-buffer-size> parameter in broker.xml.  
  • Removed extraneous '}' character from the first LDAP configuration example in section [3.14 - User Authentication Using an LDAP Server](#). |
| 15-Feb-2019 | 1.23    | CT | • Added the **Command-Line Tools** chapter along with documentation for the following command-line functions:  
  o Load KnowledgeBase  
  o Bulk Load LCD Carriers  
  o Password Encryption  
  o Configuration Saver  
  o Configuration Loader |
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>CH</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15-Feb-2019 | 1.23    | ST | - The document has been reorganized so a new Section called **Support Tools** has been added which includes Trace, Config Capture and Loader tool functionality and Performance Monitor.  
- Updated section 2.4 - **Configuration Saver and Loader Tools** in this chapter to reflect that a command line option has been added.  
- Updated screenshots to section 2.5 - **Configuration Capture/Load Via the Command-Line** in this chapter to reflect details of how to use the new command line option.  
- Build 101 also includes the following updates:  
  o Support for a command line interface for the loader and the saver.  
  o Improved performance of configurations moving into environments with large claim stores by only disabling/re-enabling constraints on the claim store tables if those tables are to be operated on by the config loader. (This is limited to cases when the enterprise structure changes and the claim store becomes invalid in its current form or claims are being imported into the system via the configloader).  
  o Improved performance in the saving of tables for the saver and via the support page within the application. |
| 21-Sep-2018 | 1.22    | ST | - Updated screenshots to reflect Build 72  
- Minor corrections throughout. |
| 10-Aug-2018 | 1.20    | TC | - Major revisions to section 5 - **Setting up Secure Communications**. Corrected many mistakes in the “overview” section and added many more detailed examples of the use of the keytool utility for import certificates into keystores and truststores. |
| 18-Jun-2018 | 1.19    | TC | - Added section 3.1.2 - JBoss Cache Configuration (5.4 SP1-CU05+).  
- Added section 3.1.3 - Enabling Garbage Collection Logging for the JBoss JVM.  
- Added section 3.10 - Cross Claim Type Editing.  
- Made corrections to section 3.11 - Claim Line Expansion. Also added a description of what the feature does.  
- Made several corrections throughout section 4 - **Multi-Node Claim Processing with High Availability** in order to reflect changes to configuration files that had been made by cumulative updates prior to 5.4 SP1-CU05.  
- Enhanced section 3.14 User Authentication Using an LDAP Server by adding notes to the “java.naming.provider.url” and “baseCtxDN” entries in the Configuration Tips section. Also clarified the “bindCredential” entry in this same section to indicate that the use of encrypted passwords is not currently possible, but that the password can be read from an external file to improve security.  
- Added a “Multi-Tenant Note” to section 2.1 - Overview of the **Support Tools** chapter to warn the user that a configuration capture from a multi-tenant system can only be imported into another multi-tenant system. |
<p>| 7-May-2018  | 1.18    | TC | - Added a specific LDAP configuration example for Active Directory in section 3.14 - User Authentication Using an LDAP Server. |
| 4-Apr-2018  | 1.17    | TC | - Added note in section 3.14 - User Authentication Using an LDAP Server for how to configure two LDAP servers. The login module template was modified to add the “name” attribute so that each module can be uniquely identified. |
| 2-Apr-2018  | 1.16    | TC | - Replaced the BLUE highlight with YELLOW highlighting in the LDAP configuration example in section 3.14 - User Authentication Using an LDAP Server. |
| 22-Mar-2018 | 1.15    | TC | - Made class name correction to the LDAP configuration. (See code snippet just above Configuration Tips in section 3.14 - User Authentication Using an LDAP Server.) |</p>
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>CH</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>25-Feb-2018</td>
<td>1.13</td>
<td>TC</td>
<td>Added note to section 3.13.3 - Patient Claim Serialization (post-SP1 CU04) explaining that claims released from the PCS patient queues are sent with HIGH priority on JMS.</td>
</tr>
<tr>
<td>22-Feb-2018</td>
<td>1.12</td>
<td>TC</td>
<td>Added section 4.2.2 - Purge: max Parameter</td>
</tr>
<tr>
<td>12-Feb-2018</td>
<td>1.11</td>
<td>TC</td>
<td>Added section 4.4 - Non-HA Configuration</td>
</tr>
</tbody>
</table>
| 30-Jan-2018 | 1.10    | TC | Edited sections 3.11 and 3.12  
• Added LDAP and PCS to Glossary                                                                                                                      |
| 19-Jan-2018 | 1.9     | TC | Added the following sections:  
• 3.13.4 - Startup Timing - Options for how the Claim Connector waits for configuration information from JBoss during startup.  
• 3.13.5 - Handling Large Claim Messages - Described properties in the jndi.properties file and how to configure to better handle large claims with patient history. |
| 8-Jan-2018  | 1.8     | QT | Added section 2 - Delete Inactive LCDs in Bulk to include instructions on how to delete inactive LCDs in bulk.                                                                                           |
| 16-Dec-2017 | 1.7     | TC | Enhanced section 3.13 - Claim Connector Configuration to include configuration of new parameters added with the Patient Claim Serialization feature.                                                        |
| 27-Sep-2017 | 1.5     | ST | Updated section 2.3 - URL/Support Page Method to include new functionality in the support page.  
• Renamed:  
  Include the following configuration information to Capture Database System Configuration.  
• Added:  
  Capture database structure information without any data  
  Include KB Load logs  
  Include configuration files used to configure the connector  
• Include configuration files used to configure application server |
| 25-Aug-2017 | All     | TC | Formatting changes only.                                                                                                                                                                                  |
| 8-Aug-2017  | 1.5     | ST | Added DDR Trace Instructions section.                                                                                                                                                                     |
| 7-Jul-2017  | 1.4     | TC | Improved the explanation of the table in section Summary of Configuration Options.  
• Added note to section 3.7 - JBoss Logging Verbosity that explains why a service restart is required after changing logging configuration.  
• Added a note to section Getting Started regarding restoring the icp.keystore file after 5.4 SP1 has been installed. |
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>CH</th>
<th>Description</th>
</tr>
</thead>
</table>
| 22-Jun-2017| 1.3     | TC | - Enhanced sections 1.1 - Summary of Configuration Options and 3.1.1 - JBoss JVM Memory Size to underscore the need for at least 6GB memory for JBoss when the DDR LCD rule is used.  
- Added more detailed instruction in section 3.2 - Claim Processing Threads for selecting the most appropriate value for the engine.count setting.  
- Correction to section 3.9 - Claim History Retrieval to note that the current default of “multi-query” is the preferred setting.  
- Added section 3.11 - Claim Line Expansion to describe how it is configured.  
- Fixed multiple problems with Figure 1 in section 4 - Multi-Node Claim Processing with High Availability.  
- Fixed multiple problems with Figure 3 – Logical System Configuration for High Availability (also in section 4). |
| 30-Mar-2017| 1.3     | TC | - Added section 4.7 - Operational Considerations.  
Updated section 4.1.2 - Configuring the broker.xml File with instructions for setting <persistence-enabled> to true. |
| 10-Dec-2016| 1.2     | ST | 9. Modifications to section 1 - DDR Trace Instructions:  
- To export the data and send to someone for review, select Export in the top right corner.  
- Select Save and save the file. The file can now be emailed or attached.  
- Updated section 1.1 - Requirements.  
- Removed section on running a trace without the February 2016 DDR UI update installed. |
| 1-Apr-2016  | 1.1     | ST | - Updated trace instructions. Changed heading of section 1.3 to ‘Reading Available Trace Data.’ |