MES Server Sizing & Architecture Guide

sepasoft

Version History

Version Number	Revision Date	Revision Description	Change By	Additional Notes
1.0	1/29/2024	Initial Release	Tom Hechtman	Includes OEE, SPC, and Traceability CPU and memory expectations.
1.1	06/25/2024	Added Test Results	Taylor Gentry	Settings & Change-over is coming. Batch is coming.
1.2	06/26/2024	Added DB connection tuning. Added DB growth time period. Other minor corrections.	Tom Hechtman	

Conventions

	Sepasoft
	Ignition
	Customer
	Important fact
	Useful tip
	Critical fact

	This guide only covers Sepasoft MES-related modules, and it is common for Ignition applications to include additional functionality that consumes resources. It is the responsibility of the integrator or customer to account for non-MES-related functionality when sizing servers.

Introduction

Sepasoft provides an MES solution for the Ignition platform and is highly configurable to meet customer requirements. This degree of configurability in Ignition and Sepasoft MES can lead to the overuse of system resources. This guide provides some best practices for implementing a performant and scalable MES solution. It is important to note that this guide only covers the MES portion, and it is common for Ignition applications to include additional functionality that consumes resources. Throughout that process, we recommend observing the server's performance characteristics to make any necessary adjustments to the architecture. There is no performance guarantee since it is based on customer design choices.

Definitions

The following terms are used when discussing MES architectures:

Enterprise

The enterprise is the highest level of a company or business unit, with a wide MES architecture involving multiple servers on-premise and in the cloud.

Site

A site is a fixed geographical production location in an enterprise. Separating your enterprise into multiple production sites allows for handling production differences between sites or performing analysis across them.

Area

A site can be divided into sections, usually by type of process. For example, receiving, preparation, processing, packaging, etc.

Architectures

The Sepasoft solution ranges from a simple single-server setup to a complex enterprise involving numerous servers across several manufacturing facilities. This section covers the various architectures and considerations.

Single Server

Single server implementations are intended for a single production facility (site) or area. It is possible for multiple sites to be handled by a single server, but the following must be considered.

- A license for each site: Even though savings can be recognized from reduced hardware and Ignition licenses, an MES license for each site is required.
- Dependency on the application during WAN service outages: If there is no tolerance for a loss in production for WAN outages, an on-premise server is recommended. Even robust fiber WAN connections can experience disruptions. Outages have occurred due to issues with major ISP routers in regional areas, sometimes lasting days.

Consider the following when determining your MES server location: Is the MES functionality passive? Production can continue if OEE is just being tracked. On the other hand, quality, inventory, schedules, workflow procedures, and SOP documents are active, and their loss will cause production stoppage. Can the loss in functionality be temporarily replaced with paper systems?

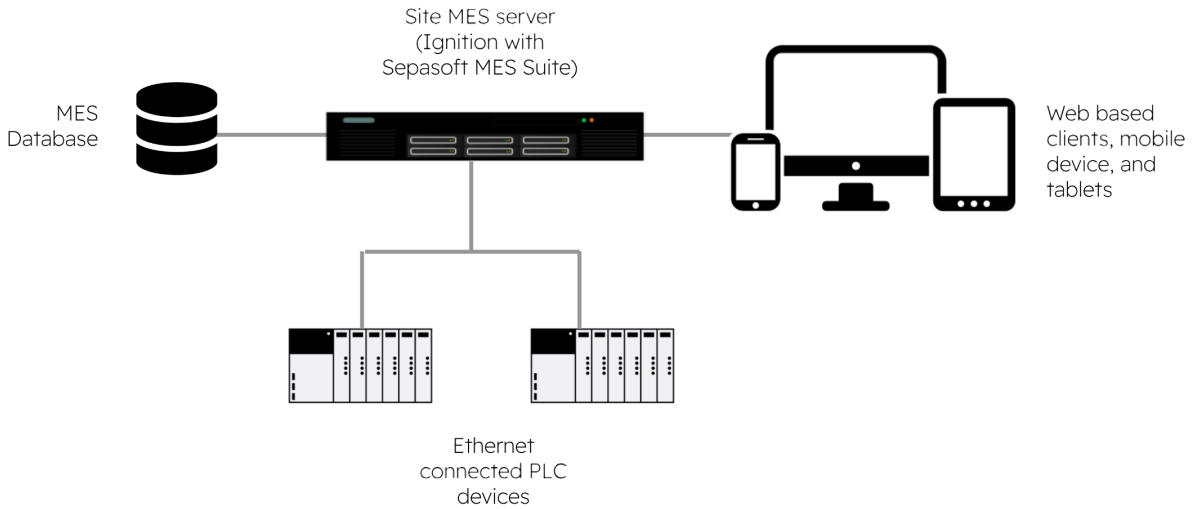
Is the production rate high, and will network latencies cause the production rate to be slower? If machines are cycling every five seconds and numerous values have to be recorded in two hundred milliseconds, then depending on a WAN connection will likely affect the maximum throughput.

- WAN bandwidth and latency: Because MES typically collects values from machinery on the plant floor, communication between the controllers and the MES server is required. A responsive MES system requires sufficient bandwidth with minimal latency.

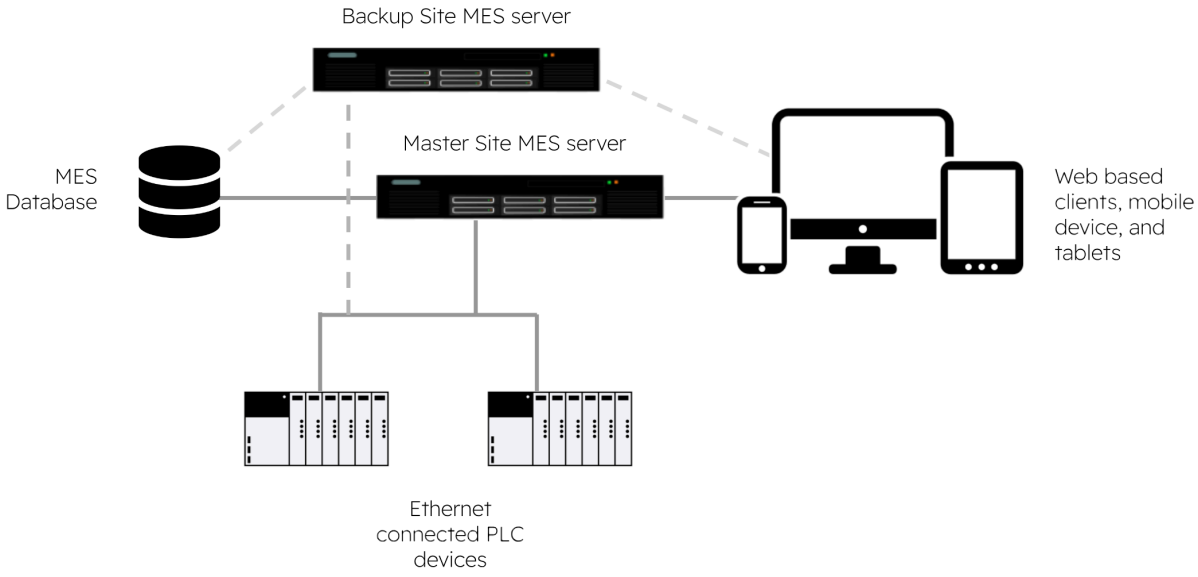
  Consider this scenario: An MES server, located either in the cloud or at a central data center, serves four different sites. If each site uses OPC UA to communicate with plant floor equipment to collect production counts, equipment states, quality information, etc., and each site also supports hundreds of user sessions, this architecture will consume considerable network bandwidth. Allocating only the typical bandwidth necessary for non-MES applications may result in a poor user experience.

  Using MQTT instead of OPC UA is recommended in this type of architecture and will help reduce bandwidth usage. Nevertheless, it is essential to ensure that adequate bandwidth is available. Unfortunately, due to the multitude of scenarios and variables involved, we are unable to provide a definitive formula for determining the bandwidth requirements.

- Server load: Depending on application and equipment items, server loading must be considered. Because running MES on the Ignition platform is not elastic (meaning more servers can be spun up to handle peak loads), **limiting an MES server to one hundred equipment items is recommended.** This general best practice can vary depending on the use case. For example, if each line is a single machine (machine shop use case), then one server can handle more lines. If each line has two hundred machines (equipment manufacturing line use case), then one server should handle fewer lines.



BASIC SINGLE-SERVER ARCHITECTURE



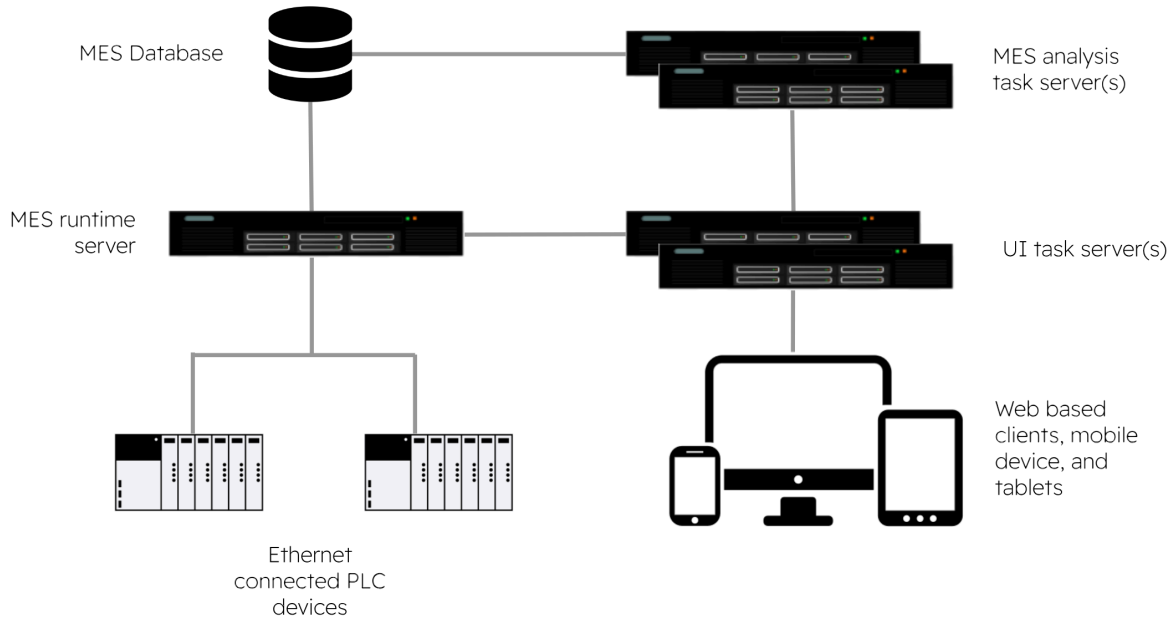
SINGLE SERVER WITH REDUNDANCY ARCHITECTURE

Multi Server

Multiple server implementations are intended for production facilities with a large amount of equipment, high production rates, heavy analysis loads, or other functionality where the distribution of processing load is required.

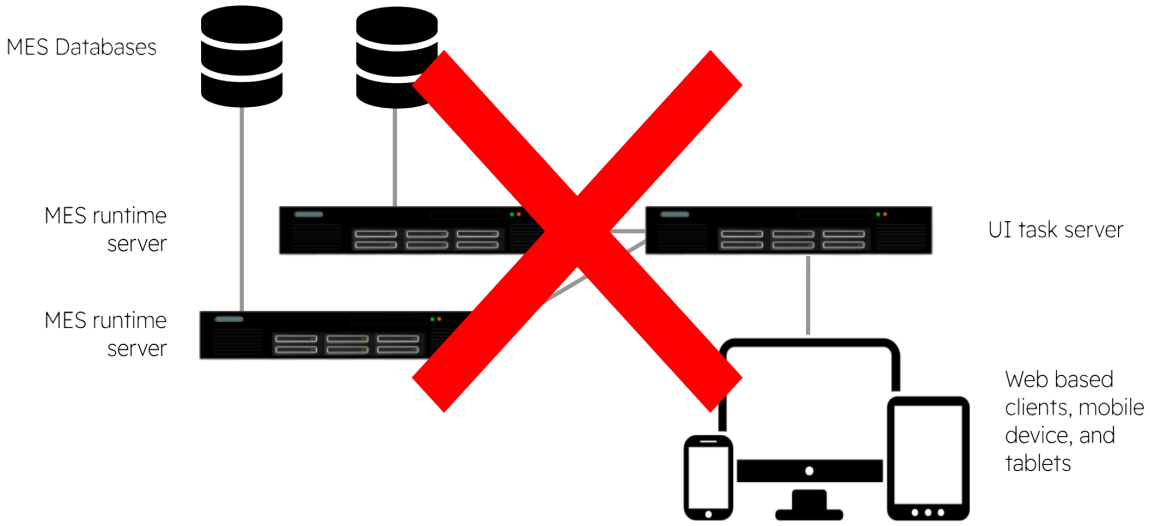
Sepasoft MES modules have a task routing feature for offloading user sessions or analysis to other Ignition servers. The following should be considered when using task routing:

- MES licenses are only needed for the runtime servers controlling or collecting live production data. For example, monitoring equipment state, production counts, recording trace information, running batch recipes, executing production workflows, etc., all require MES licenses. User interfaces and MES analysis tasks can be offloaded to one or more Ignition servers without purchasing additional MES licenses. To enable task routing, Sepasoft must configure the relevant licenses.
- All MES servers handling user interface sessions or MES analysis should be located on the same LAN.



MES ARCHITECTURE WITH TASK ROUTING

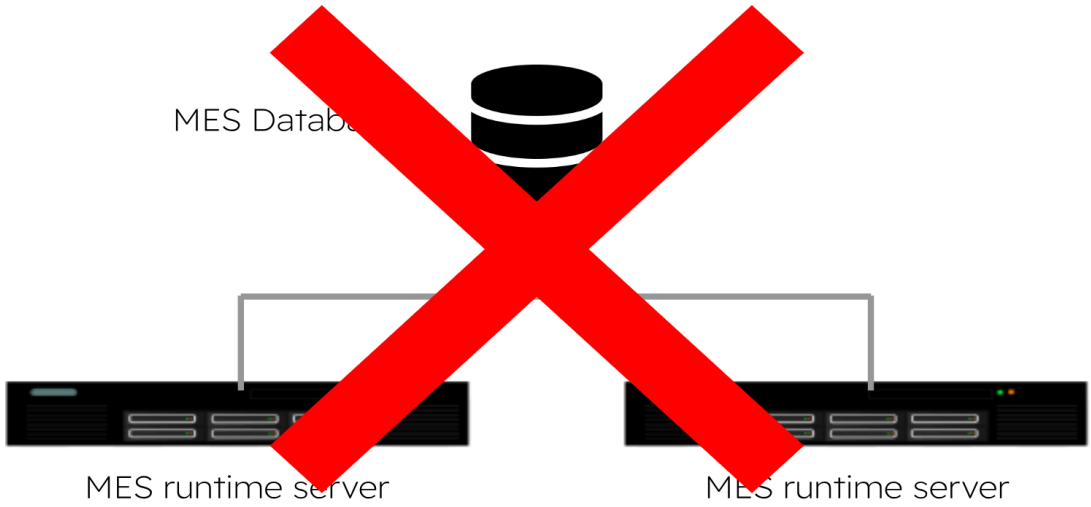
- UI task routing servers can only handle frontend user requests for a single MES runtime server. Likewise, analysis task routing servers can only be connected to a single MES database instance and perform analysis for a single MES runtime server.



MES ARCHITECTURE WITH INVALID TASK ROUTING

	Task routing servers can only support a single MES runtime server.

- Only one MES runtime server can control each MES database instance. The MES runtime server will create and update the schema and is the only one to write to the tables. Analysis task routing servers are set up to only read from the MES database schema.



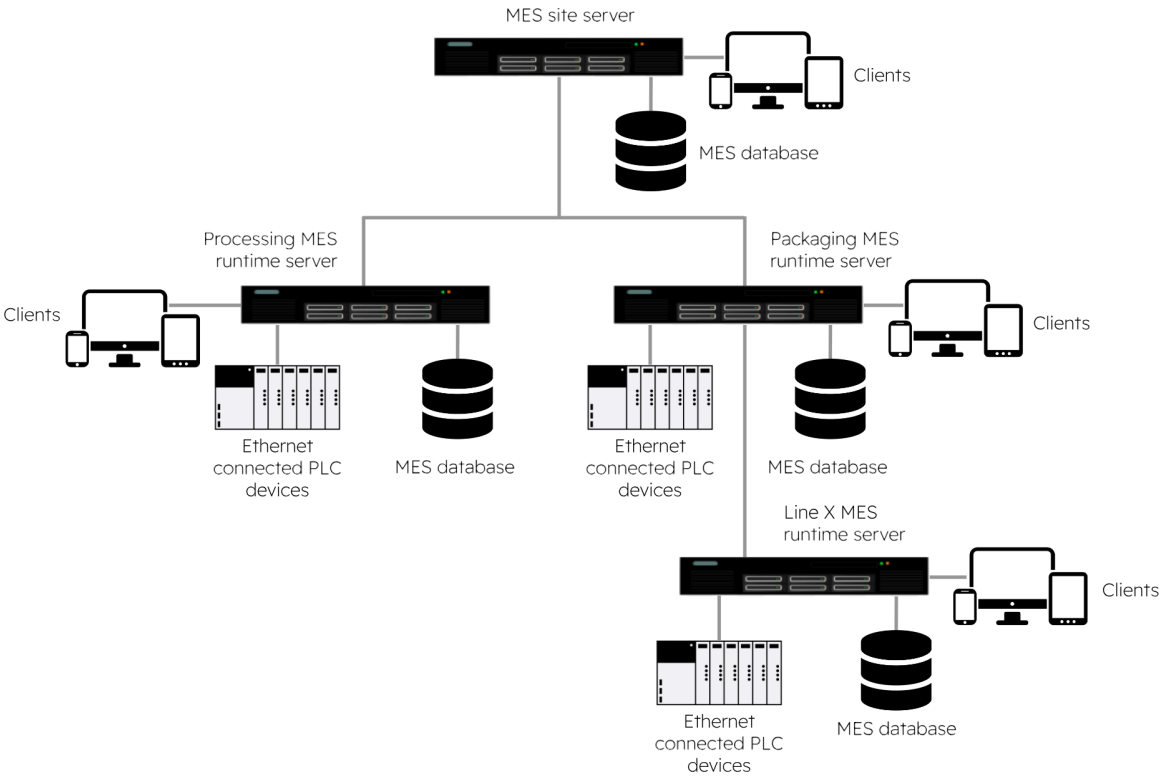
INVALID MES ARCHITECTURE

	Never connect multiple MES runtme servers to the same database instance. Redundant Ignition servers without proper licensing may produce the same duplicate database usage error.

	Attaching multiple MES runtime servers to a single MES database instance can happen unintentionally by not changing the database connection URL for different environments like development, testing, or production.

In addition to task routing, multiple servers can handle specific production areas, lines, or process cells within a site. This allows for the distribution of the load across multiple MES servers or limits the impact in the event of server failure.

- Based on the configuration, the production data can be shared between MES servers for the site.

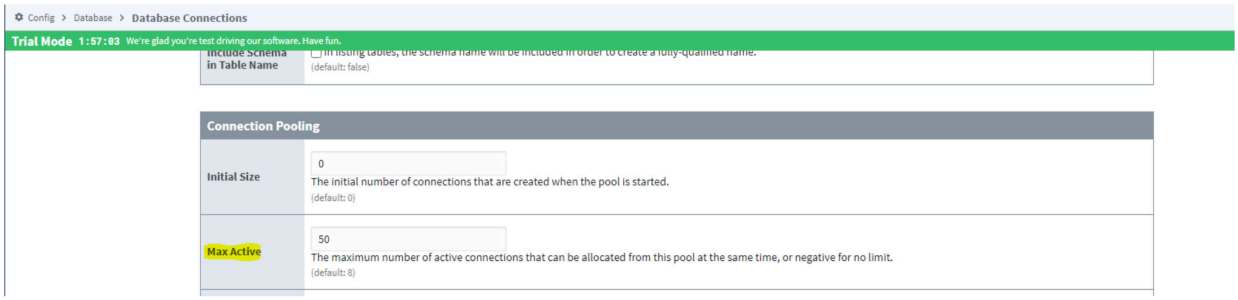


MES ARCHITECTURE WITH MULTIPLE SERVERS BELOW SITE

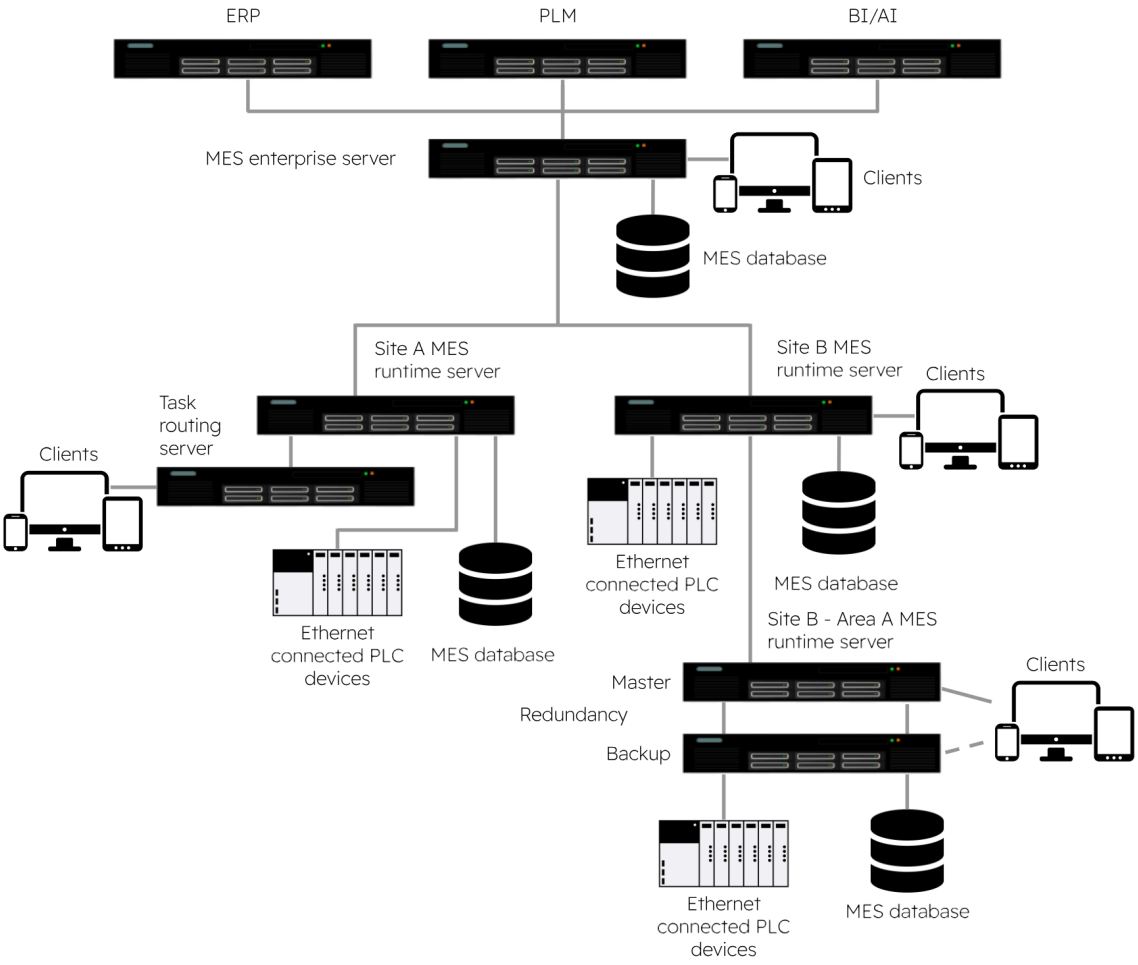
Enterprise

The MES enterprise functionality is intended for business units with multiple production facilities. Built-in functionality handles synchronizing materials, batch recipes, schedules, production workflows, documents, etc., and is automatically synchronized between the appropriate MES servers of the MES enterprise.

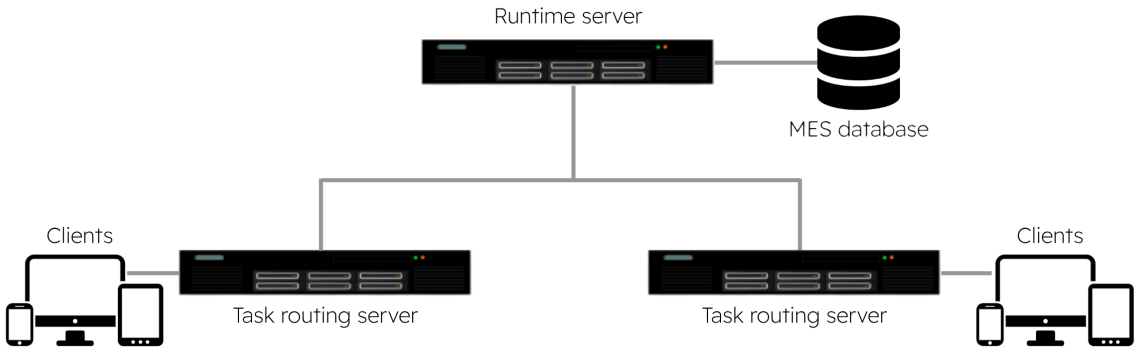
- Only one enterprise node is supported in the equipment model. Business units within a company that do not want to share materials, schedules, production workflows, batch recipes, WIP inventory, etc., need separate MES architectures and associated licensing.
- The synchronization requires the configuration of multiple Ignition Gateway Network connections between servers.
- For each child server, at least one database connection must exist. For example, if a root MES Enterprise server instance has fifty sites beneath, the MES database must be configured for at least fifty connections. It is not recommended to set this to unlimited. Additionally, the database server must be configured and licensed to handle fifty connections or more at all times.If there are multiple levels, the same applies. For example, if an MES site has four area servers beneath it, four database connections must be allowed on the site server.



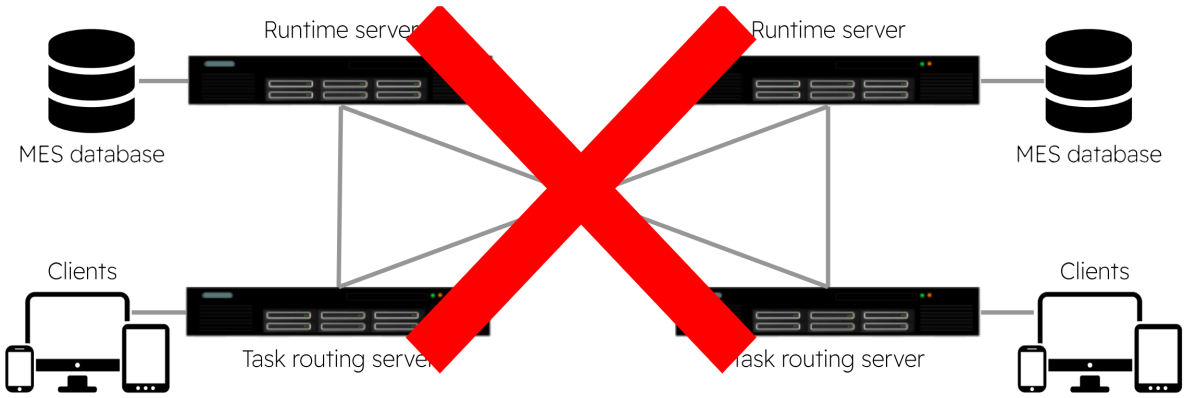
IGNITION MES DATASOURCE WITH THE MAX ACTIVE CONNECTIONS SET TO 50



MES ENTERPRISE ARCHITECTURE WITH MULTIPLE SITES



MES TASK ROUTING WITH USER-INTERFACE SERVERS



INVALID MES TASK ROUTING ARCHITECTURE

⚠	For MES enterprise architectures, a database connection must be allowed for each child MES server.

Server Sizing

Processors

Ignition and the MES modules multitask and depend on threads to handle the numerous tasks that occur simultaneously. More processors and cores to service these tasks generally result in a better user experience during peak loads.

Capitalizing on all the benefits of application-sharing platforms is commonplace for Ignition instances. However, this means that the processors and cores must be shared among applications outside of a single Ignition instance. If other applications have peaks consuming processors, it can cause poor performance of Ignition. Often, this is managed by IT and is not transparent to staff working with Ignition. Keeping this in mind and using client-server best practices will reduce the processor demand needed.

Disks

Hard disk space is likely the least important of all the server resources Ignition requires. Although minimal, there does need to be enough to store the Ignition projects and logs.

Common MES Server Configurations

This guide only covers Sepasoft MES-related modules, but it is common for Ignition applications to include additional functionality that consumes resources. It is the responsibility of the integrator or customer to account for the non-MES-related functionality when sizing servers.

This guide only covers Sepasoft MES-related modules, but it is common for Ignition applications to include additional functionality that consumes resources. It is the responsibility of the integrator or customer to account for non-MES-related functionality when sizing servers.

The amount of database disk storage required depends on the MES activity and desired retention time. The Sepasoft MES modules have a setting for Data Retention Period in Days that allows for purging older production data.

Test Environment

The following hardware was used to perform load testing:

- Docker containers were used to limit the number of cores and memory for each test environment.
- Microsoft SQL using an SSD (Single schema dedicated to this test).

Server Configuration

Our servers were only executing Ignition. No other programs, virus software, or endpoint protection were executing during the testing. The database server only contained one MES schema and no other schemas.

Test Approach

Because each MES system will use different MES modules, the results are itemized to show the impact of each. This breakdown allows for a more accurate estimation of system requirements tailored to specific use cases.

The results for each MES module are further broken down into two sections:

- Core MES collects data, monitors production, and performs live analysis. This is the baseline with zero clients.
- Client simulation using JMeter to establish client Perspective sessions and generate requests to the Ignition server. Each client session simulates either a dashboard representing a heavier load or an operator control screen.

OEE System

The base OEE test system consists of the following:

- Ten lines, each with 14 cells and two cell groups, use key reason downtime detection.
- Each line is configured with the following live analysis:
  - Set for a 60-second update, Start of Run time period, and 35 data items.
  - Set for a 60-second update, Shift time period, and 35 data items.
- Each line is configured with five additional factors.
- Python script running on a gateway timer to generate downtime events involving multiple cells every 15 seconds, update three production counters per line every 60 seconds, and change additional factor values every 60 seconds.

Clients used for testing consist of the following Perspective pages:

- Run control page containing the Run Control, Downtime Table, and Timing Chart components, each set to show the last eight hours of production.
- The OEE analysis is performed on the gateway, and the results are written to a dataset tag that all clients use to present in the UI. Each line has four analysis settings that are being executed every 15 seconds.
- Scaled in blocks of four clients, including one client for a run control page and three clients using dashboard pages.

Load	Server (Ignition)	Zero Clients		With Clients		DB Growth (per hour)
		CPU	Memory	CPU	Memory	KBytes
1 line, 4 clients	4 core, 8 GB	2%	3000 MB	9%	4000 MB	525
5 lines, 20 clients	6 core, 12 GB	6%	3000 MB	30%	6000 MB	2625
10 lines, 40 clients	8 core, 16 GB	7%	3000 MB	35%	11000 MB	5250

SPC System

The base SPC test system consists of the following per Location where samples are collected:

- Two sample definitions with ten attributes and five measurements.
- The location is configured with Shift period live analysis on a 60 second update rate including:
  - Signal Out of Control for two sample definitions.
  - XBar, Range, and Standard Deviation for each attribute of both sample definitions.
- Python script running on a gateway timer to generate samples every 60 seconds containing all attributes and measurements (100 values).

Each SPC client used for testing consists of the following Perspective pages:

- Control Chart page containing four control charts showing the past eight hours of data.
- The charts are updated every 60 seconds when new samples are collected.
- Scaled in blocks of four control chart pages.

Load	Server (Ignition)	Zero Clients		With Clients		DB Growth (per hour)
		CPU	Memory	CPU	Memory	KBytes
1 location, 4 clients	4 core, 8 GB	1%	2000 MB	2%	2000 MB	21150
5 locations, 20 clients	6 core, 12 GB	1%	3000 MB	4%	2000 MB	141300
10 locations, 40 clients	8 core, 16 GB	2%	6000 MB	5%	4000 MB	372800

Track & Trace System

The base Track & Trace test system consists of the following:

- Material is moved through 10 cells of the line structure used for OEE. The line structure was reused, but it can be any storage unit, cell, line, unit, etc.
- The lot at each cell is indexed to the next cell every 60 seconds.

Clients used for testing consist of the following Perspective pages:

- Trace Graph page cycling through lots on a 60-second cycle.
- The inventory screen shows the current inventory for the first and last cells. The inventory screen updates every 60 seconds.
- Total of four clients per line—two for the trace graph and two clients for inventory.

Load	Server (Ignition)	Zero Clients		With Clients		DB Growth (per hour)
		CPU	Memory	CPU	Memory	KBytes
1 line, 4 clients	4 core, 8 GB	1%	2000 MB	1%	2000 MB	16600
5 lines, 20 clients	6 core, 12 GB	1%	2000 MB	1%	2000 MB	89750
10 lines, 40 clients	8 core, 16 GB	1%	4000 MB	1%	2000 MB	199250

Settings & Changeover System

Coming soon.

The base Settings and Changeover test system consists of the following:

- One-hundred-fifty setting values are added to each of the 14 cells of the line structure used for OEE. The line structure was reused, but it can be any storage unit, cell, line, unit, etc.
- Two Settings and Changeover recipes are toggled between every 15 minutes.
- Twenty-five tag values per cell are changed to cause value variances to be recorded every minute.
- Values are only written to the Ignition tag. The time for the values to appear in the PLC depends on the model of the PLC, communication method, and bandwidth.

Clients used for testing consist of the following Perspective pages:

- Recipe value variance page that contains the Settings Variance Viewer component showing variances for the production run and selected equipment item.

Load	Server (Ignition)	Zero Clients		With Clients		DB Growth (per hour)
		CPU	Memory	CPU	Memory	KBytes
1 line, 4 clients	4 core, 8 GB					
5 lines, 20 clients	6 core, 12 GB					
10 lines, 40 clients	8 core, 16 GB					

Batch Procedure (Workflow) System

Coming soon.

Load	Server (Ignition)	Zero Clients		With Clients		DB Growth (per hour)
		CPU	Memory	CPU	Memory	KBytes
1 line, 4 clients	4 core, 8 GB					
5 lines, 20 clients	6 core, 12 GB					
10 lines, 40 clients	8 core, 16 GB					

OEE, SPC, and Track &Trace Combined

These results reflect the combination of OEE, SPC, and Track & Trace. The test conditions were the same as detailed for each module above.

Load	Server (Ignition)	Zero Clients		With Clients		DB Growth (per hour)
		CPU	Memory	CPU	Memory	KBytes
1 line, 12 clients	4 core, 8 GB	2%	3000 MB	12%	3000 MB	38100
5 lines, 60 clients	6 core, 12 GB	7%	3000 MB	31%	10000 MB	213150
10 lines, 120 clients	8 core, 16 GB	8%	5000 MB	37%	14000 MB	451150

Best Practices

Ignition is very flexible and allows for numerous methods of implementing applications, enabling engineers to easily achieve customer requirements. However, this versatility also means that it is possible to use inefficient techniques.

Memory

Besides the amount of server memory, several memory-related considerations can vastly affect server performance.

- Physical versus Virtual Memory. Physical memory is the amount installed into a computer or allocated to a VM or container. Virtual memory is beyond physical memory when the operating system saves a page of physical memory to the disk, making the physical memory available to new allocation requests. This allows programs to use more memory than is physically available. The operating system controls whether virtual memory is being used, and most programs can handle this without a noticeable impact on performance. Because Ignition is a real-time system frequently accessing values stored in memory, performance is greatly degraded once the memory allocated to Ignition starts being swapped out to disk.

For MES applications, ensuring sufficient physical memory for Ignition is important.

Never set the Initial Java Heap Size and the Maximum Java Heap Size properties in the ignition.conf to a value more than the available physical memory.

It is recommended to set the Initial Java Heap Size and the Maximum Java Heap Size properties in the ignition.conf to the same values. This will cause Ignition to allocate all memory during startup and prevent memory allocation errors during production.

Example scenario:

Memory required for the operating system: 4GB
Memory assigned to Ignition: 16GB

Ignition config settings

```
# Initial Java Heap Size (in MB)
   wrapper.java.initmemory=16384

# Maximum Java Heap Size (in MB)
    wrapper.java.maxmemory=16384
```

Total Physical Memory Installed: 20GB

When running Ignition on a Virtual Machine (VM) or container, ensure the assigned memory is covered by physical memory, and not other VMs, because containers running on the same server can reduce the amount of needed physical memory required for Ignition.

- Amount of Memory. By default, Ignition is configured for 2GB of memory, and in the typical MES implementation, more is required. See the Common MES Server Configurations section for more details.

- Garbage Collection. Once a block of memory is allocated from the operating system, the program (Java in the case of Ignition) handles the granular details of values stored in the memory. When storage of a value is no longer needed, the garbage collector will free up the memory so other values can be stored. By default, Ignition uses the G1GC garbage collection algorithm. Changing the settings can cause garbage collection to happen less frequently and cause brief pauses in Ignition while garbage collecting larger blocks of memory. Using the default garbage collection settings is recommended unless directed otherwise by Inductive Automation.

Module optimization cannot fix an inefficient Ignition implementation.

Database

The MES modules store configuration and production data in a database. The performance of the database has a major impact on the overall performance of the MES application.

- There are four database vendors supported by the Sepasoft MES modules:
  - Microsoft SQL Server
  - MySQL
  - Oracle
  - PostgreSQL
- Use A Dedicated Database Server. The SQL database should be on its own dedicated server separate from the Ignition server. Because the peak loads of Ignition can request more database queries, the demand for system resources will happen simultaneously. Distribute this load over two servers for best performance instead of just one.

For best results, use an SSD hard drive for your database server.

- Database Performance and Capacity. Because MES depends on frequent and just-in-time database queries, the poor performance of the database server will be reflected in the MES user experience. It is common to have multiple database schemas on the same database server instance, which means heavy, cyclic, or occasional peaks of other databases can affect the MES database.

The Sepasoft MES modules don't use a proprietary database vendor and instead use the database vendor of the customer's choice. This allows them to capitalize on following company standards that align with their knowledge, existing infrastructure, backup procedures, support tools, etc. This also means that the customers or integrators are responsible for managing the performance of their database instance.

Understanding the data flow for select queries and its impact on system resources is important.

Select query data flow:

1. A Select statement is initiated on Ignition. The initiation can be from MES, a named query, or scripting. All have the same data flow.
2. The database manages the execution of the query and begins streaming the results back to Ignition. This requires network bandwidth.
3. Ignition receives the streamed results and stores all rows in memory. This requires Java to allocate memory and consumes space in the Ignition heap memory until it is no longer needed.

Because the results are stored in memory, it is possible to fault or significantly impact the performance of the Ignition server. Normally, initiating such a query is not intentional but can happen with uninitialized query parameters or data ranges saved in project resources.

The MES modules depend heavily on the database; an undersized database will adversely affect the MES performance. The major items to consider when ensuring the database has enough capacity are:

- Maximum connections
- Maximum disk storage capacity
- Server memory for optimum performance
- Server CPU for optimum performance

Always install the MES database and Ignition on separate servers.

- Use An Isolated MES Schema. Configure a separate schema for the MES module. Other schemas can be created on the same database instance, provided they don't create too much demand on the database server.

  The Sepasoft MES modules automatically create the database tables and indexes used by the modules. It is best to add any custom tables to a different schema. This will simplify database tasks like backup, restore, and other maintenance tasks. Additionally, the data source configured in Ignition will only be used by the MES modules, preventing inadvertent starvation of database connections and adversely affecting performance.

Always use a separate database schema for MES.

For Microsoft SQL Server, use the Read Committed Snapshot isolation level. This will keep reads from blocking writes and writes from blocking reads, and increase query throughput.

When using Microsoft SQL Server, it is recommended to verify the isolation level, and modify if needed. It can be set for each database of the server instance. To check the current isolation level setting for the MES database, use Microsoft SQL Server Management Studio to execute the command below.

```
use MES_DB_NAME
dbcc useroptions
```

If the results list the isolation level as **read committed**, then changing it using the following commands is recommended.

```
alter database MES_DB_NAME
set ALLOW_SNAPSHOT_ISOLATION ON

alter database MES_DB_NAME
set READ_COMMITTED_SNAPSHOT ON
```

- Stop Production Before A Scheduled Database Outage. Because the MES system stores configuration and production data in the database, it cannot continue production while the database is unavailable. Because many MES functions require saving and reading current values to and from the database, store-and-forward technologies cannot be used with MES.

  To prevent loss of data or interruption of production, it is important to stop active production runs in the MES system before shutting down the database.

- Don't Access MES Tables Directly. Writing SQL queries or altering structures for the tables managed by the MES modules is tempting. This can cause adverse performance issues and should be avoided. For more detailed information, read Accessing Sepasoft MES Database Tables.

Resist the temptation to access information directly from the MES database tables. If you feel you need to access the MES tables, please consult with Sepasoft first. We have likely encountered your use case and will have a solution for you.

- **Number of Database Connections.** The MES modules will make multiple database queries simultaneously, and too few connections can negatively affect the performance of the MES application. The Ignition data source configuration allows setting the maximum number of connections. Even more important is to ensure your database can handle the number of connections. This includes all other connections to the database by Ignition or other applications. Separate from the database, the operating system may limit the number of connections.

  In the case of MySQL 5.5, which can handle 151 connections by default, the following scenario can lead to connection starvation:

  MES data source: 20
  (maximum connections)

  Other Ignition data sources: 20
  (maximum connections)

  Other applications other than Ignition: 120
  (maximum connections)

  Total number of connections needed: 160

When the MES database connection is configured in Ignition, it defaults to eight simultaneous connections. Most MES projects require Increasing this value to accommodate peak loads.

- **Tune the Database Connection Pool Size.** By default, an Ignition database connection pool size is eight. Because MES relies on several database requests at any given time that varies with production activities, this pool size must be adjusted. Use the following procedure to tune the database connection pool size as production activities increase.
  - View the number of active connections on the Ignition Database Connection Status page.
  - If the number of connections is over 80% of the connection pool setting, increase the number in the advanced database connection setting section.
- **Connection Latency.** The number of database queries per second for MES applications can be high, and any latency in the connection to the database can significantly affect the performance. The source of latency can be the remote database server, a firewall, a slow network connection, an undersized database server, etc. Consider a latency of 10 milliseconds with 200 queries per second; 10 * 200 is 2,000 milliseconds or 2 seconds. Since 2 seconds is greater than 1 second, not all the queries can be completed in a timely manner.

Use Ignition database stats to monitor the number of connections, query throughput, etc.

In the Ignition gateway configuration, set the **Test on Borrow** option to false for the MES database connection.

- **Database Maintenance and Backups.** There are many different approaches regarding backing up databases, and guidance will not be covered in this document. However, that doesn't mean that backing up your database is unimportant. It is not enough to routinely back up the database. It is also important to verify that the backups are valid and to do test restores occasionally.

  In addition to backups, maintenance, such as packing, check indexes, etc., should be done routinely.

Tags

The MES modules capitalize on the power of tags. Tags are used to expose internal MES data to users and are also used to communicate with field devices such as PLCs. The following are considerations related to using tags:

- **Standard Tag Provider Limitations.** Depending on the number of tags, the amount of data stored in the tags, and how frequently tag values change, the built-in standard tag provider might be stressed. The built-in standard tag provider stores tags, including their values, in the Ignition internal database. This means more tags, value changes, and large tag values consume the internal database using a single connection.

  Alternatively, data can be stored in custom database tables for temporary storage.

- **Scan Classes.** Tag groups (or scan classes) dictate how often Ignition polls data from PLCs. Make sure you are using proper polling rates. Not everything has to be at a one-second rate. Some values can be polled slower than others since they don't change very often. Try to determine all of the possible polling rates you require.

Screen Design

The MES modules capitalize on the flexibility of the Ignition platform. This empowers the creation of applications that meet the customer's requirements, which are impossible with many other HMI/SCADA/MES systems. With this unleashed power comes greater responsibility of the application developer to utilize good client-server techniques. For example, you may have noticed that the Amazon website only shows 20 items to the user at a time. Showing more than is useful to the user wastes system resources. As an application developer, you must be mindful of system resources used for every screen, view, or script created.

Utilize good industry-standard client-server techniques when designing Vision screens or Perspective views.

- **Use Analysis Controller Sparingly.** Adding Analysis Controller components to Vision screens or Perspective views is easy, but it is not scalable as more users are added.

The following are alternatives for a performant client-server application (which is the Ignition platform model):

- Use Live Analysis. A live analysis item is updated when relevant data changes or every 60 seconds if data doesn't change, and the results are exposed in Ignition tags. For example, the current downtime reason can be displayed, and it will immediately update the tag when it changes. The tags can be displayed on Vision screens or Perspective views and are only updated on value changes. This approach results in a better user experience and less load on the system compared to every client asking for the current downtime reason every second.
- Perform Analysis in the Gateway Context. Live analysis doesn't support multi-row results; however, analysis can still be executed once, and results can be saved in tags. Tag value change events can be used to trigger the recalculation. Tags set to data type Dataset can hold multi-row results, and when they change, Vision or Perspective clients binding to the tag will automatically be updated.

Eliminate the use of timers for polling of MES functions. If you feel you need a timer to poll MES functions, please consult with Sepasoft first.

Upgrading

Always make a gateway and database backup before upgrading. MES database table structures, data formats, or indexes are commonly altered during upgrades, and we only support forward compatibility.

Track & Trace

Initially, Sepasoft strictly followed the ISA-95 specification, which calls for defining operations with specific details about the materials, personnel, and equipment. Because this approach is tedious and inflexible in handling many production scenarios, Sepasoft pivoted to a different approach. The new approach allows recording traceability information without defining operations in advance. For workflows requiring stringent controls, Sepasoft has integrated the Batch Procedure and Track & Trace modules to provide a visual, flexible, and no-code solution, ensuring efficiency and ease of use.

Don't use the ISA-95 operations to record traceability information. Instead, use the record operation script functions or integrated Batch Procedure Module features.

Task Routing

To offload the user interface or analysis computing load to different servers, Sepasoft has a task routing feature. When task routing is used, it is essential to remember that there is only one runtime server monitoring and controlling production for any given equipment. For example, two runtime servers cannot control and monitor Line 1 of production site 1.

The following are key factors when using task routing:

- Task routing servers can only communicate to a single runtime server.
- Build script libraries on the runtime server.
- Use Ignition gateway messaging to route from visualization servers to the runtime server.

Cloud Computing

Hosting Ignition with the Sepasoft module in the cloud is fully supported. There are some considerations to factor in.

- The Ignition Designer will load slower.
- Use Perspective clients instead of the Vision Client.
- Data collection methods:
  - The cloud version of Ignition doesn't support using OPC-UA. If you install Ignition yourself, you can use OPC-UA, but it is not the best option. Using OPC-UA with the native device drivers provided by Inductive Automation uses polling, increasing the bandwidth used. It also requires opening communication ports, which is not considered a secure practice.
  - MQTT is the desired method of data transfer to keep bandwidth low and provide greater security.

Miscellaneous

- Java monitoring tools can be valuable but negatively impact performance and should be used sparingly. Any tool that tracks, records, communicates, etc., is not free and will require more computing resources.
- Ignition has great logging capability, allowing engineers to add logging from within scripts. However, all great things come at a price, and this is no exception. Adding a lot of logging can create a noticeable load on the system. The processor must queue the log entries, write them to disk, and handle truncating the logs to prevent the consumption of too much disk space. It also causes spamming of the logs, which removes important log entries when the logs are truncated.

Spamming the log can be minimized by using the logger `isDebugEnabled()` and `isTraceEnabled()` functions to condition when messages are logged. The image below shows how this is done in a tag change script. The logger must be set to Debug or Trace for messages to be logged.

Script

```
1  def valueChanged(tag, tagPath, previousValue, currentValue, initialChange, missedEvents):
2      log = system.util.getLogger('CustomLogger')
3      if initialChange:
4          log.info('Start custom logger')
5      elif log.isDebugEnabled():
6          log.debug('This is a custom message to log.')
```

Always remove unimportant custom logging after debugging is complete to avoid spamming the logs.

- Handling Large Recipes. By default, Ignition/Jetty sets a maximum message size of 2048 KB when using a Perspective Session. If a Batch Recipe is too large and validation/saving is attempted, a MessageTooLargeException warning throws to the logs, and the WebSocket disconnects from the session. The session stays open and attempts to validate/save the Recipe indefinitely until you close and re-open the session.

  If the customer will be creating their own recipes, then proactively add the following statement to the ignition.conf file. For extremely large recipes, the value might need to be increased.

  ```
  wrapper.java.additional.X=-Dperspective.websocket.max-message-size=4096
  ```

Scripts using MES functions triggered by tag value change events, timers, or other places that can execute before the MES system is running should be conditioned with `system.mes.isProductionStarted()`