



InformaticaTM

Sizing and Deployment

Informatica MDM - Product 360

Version: 10.5 HotFix 3 SP 1

Table of Contents

1	Objective.....	5
2	Module Overview.....	6
2.1	Application Server.....	6
2.2	Control Center.....	6
2.3	Desktop Client.....	7
2.4	Web and REST Clients.....	7
2.5	Informatica BPM.....	7
2.6	Physical File Share.....	7
2.7	Database.....	7
2.8	Elasticsearch.....	7
2.9	Message Queues.....	8
3	Resource Limitations.....	8
3.1	CPU and CPU Cores.....	8
3.2	Memory.....	8
3.2.1	Multi-Server considerations.....	8
3.2.2	Caches.....	8
3.2.2.1	Status Cache.....	9
3.2.2.2	ListModel Cache.....	9
3.2.2.3	ListProcessor Cache.....	9
3.2.2.4	Proxy Cache.....	9
3.2.2.5	DetailModel Cache.....	9
3.2.2.6	Dashboard Component Data Cache.....	10
3.3	Network.....	10
4	Scaling Factors.....	10
4.1	Users.....	10
4.1.1	Licensed Users.....	11
4.1.2	Concurrent users.....	11
4.1.3	Concurrent Supplier Portal Users.....	11
4.2	Product, Variants, Items.....	11

4.3	Jobs	11
4.3.1	Import.....	11
4.3.1.1	Phase 1.....	11
4.3.1.2	Phase 2.....	12
4.3.2	Merge	12
4.3.3	Export	12
4.3.4	Data Quality.....	12
4.4	3rd Party	12
4.5	Workflows (BPM)	13
5	Logical Deployment.....	14
5.1	Single Server.....	14
5.2	Separate Jobs from Clients	15
5.3	Separate Servers by Role.....	16
5.4	Scale Out	17
6	Physical Deployment	17
6.1	Roles	17
6.2	Typical Deployment.....	18
6.3	High Availability with Load Balancing	19
6.3.1	Application Server with User Role.....	21
6.3.1.1	Load Balancing.....	21
6.3.1.2	Fail-over.....	21
6.3.2	Application Server with Job Role	21
6.3.2.1	Load Balancing.....	21
6.3.2.2	Fail over	21
6.3.3	Application Server with MQ Consumer Role.....	21
6.3.3.1	Load Balancing.....	21
6.3.3.2	Fail over	22
6.3.4	Application Server with Web or Service API Clients	22
6.3.4.1	Load Balancing.....	22
6.3.4.2	Fail-over.....	22
6.3.5	Media Manager File Server.....	22
6.3.5.1	Load Balancing / Fail over	22
6.3.6	Media Manager Web Server	23

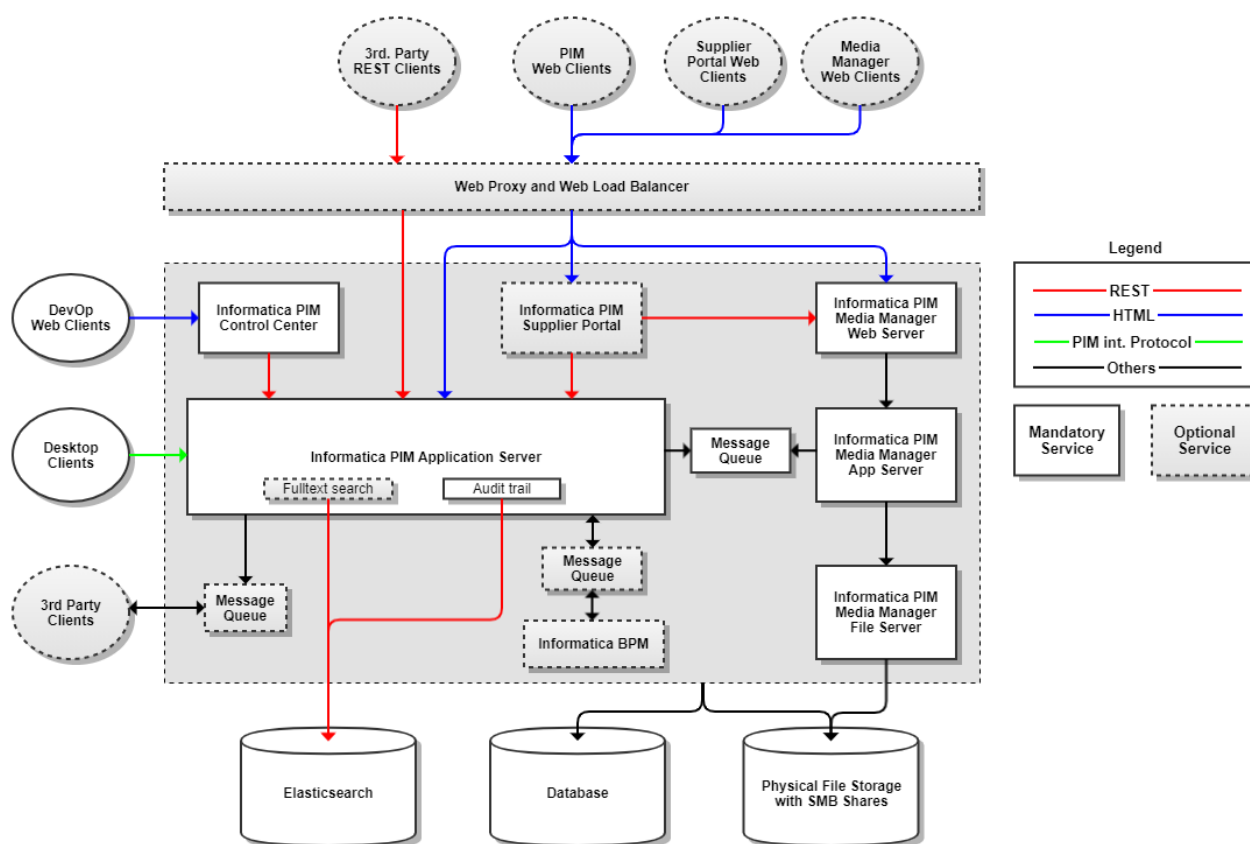
6.3.6.1	Load Balancing.....	23
6.3.6.2	Fail-over.....	23
6.3.7	Media Manager Server	23
6.3.8	Database Server	23
6.3.9	Supplier Portal Server.....	23
6.3.9.1	Load Balancing.....	23
6.3.9.2	Fail-over.....	24
6.3.9.3	Configuration	24
6.3.10	Informatica BPM Server	24
6.3.10.1	Fail-over.....	24
6.3.11	Message Queue Server.....	24
6.4	High Availability without Load Balancing.....	24
6.4.1	Application Server Cluster.....	25
6.4.2	Media Manager and Supplier Portal.....	25
6.4.3	Media Manager File Server.....	26
6.4.4	Database Server	26
7	Limitations	26
7.1	Number of Servers in the P360 Server cluster	26
7.2	Database Performance	26
7.3	Elasticsearch Performance	26

The sizing guide provides general knowledge on the sizing criteria of the Informatica MDM - Product 360 application. It contains an overview on the available modules, the needed resources and the deployment possibilities. To get a general idea on the needed resources for a specific customer scenario we provide a Sizing Calculator.

1 Objective

Informatica MDM - Product 360 is the central product information platform for e-commerce and multi-channel commerce. The solution provides distributors and manufacturers with a central solution for all product data in a central data source for the purpose of distribution in all communication channels and languages. Key elements of the solution include the import (Onboarding) and central mastering of highly structured product data and the corresponding media assets. This means that centrally managed product data can be adapted to suit the needs of various target systems (multi-channel), such as online shops or print systems, when it is exported in data formats based on CSV or XML. Product 360 comprises the following three functional areas along the supply chain from on-boarding of product data to the multi-channel export into various target systems.

2 Module Overview



2.1 Application Server

The application server of Informatica MDM - Product 360 is the highly extensible core of the application. It provides all product data management functionalities for different clients and executes the majority of the load of the system. It performs imports, merges, exports as well as other mass operations like search & replace. It is highly optimized for multi-core systems, provides several caching technologies to reduce the load on the database and provides a powerful REST based api which can be used to integrate Product 360 with other 3rd party applications.

2.2 Control Center

The control center is a small, lightweight application who is able to control and manage the application server. It is especially useful in a multi-server deployment scenario where managing multiple application servers might be cumbersome. It provides general management functionalities like start and stop of servers as well as typical operational functions. The control center is a separate application and an own service in the host machine. It's must be installed on each host which has an application server running. Additionally to the build-in management functionalities it also provides access to it's data by the SNMP protocol.

2.3 Desktop Client

The Informatica MDM - Product 360 Desktop Client is a desktop application for Windows. It provides a very flexible UI and a powerful editing interface which rich mass data editing functionalities. The Desktop Client is directly connected to one of the application servers using a proprietary protocol which is implemented directly on the low-level TCP/IP sockets from Java. The Desktop Client is designed and implemented as intranet application and needs a reasonable amount of bandwidth. Mass data handling after all transmits a lot of data.

2.4 Web and REST Clients

All HTTP based clients should be connected to a reverse proxy for security purposes. This proxy can also provide a load balancing logic in case of a multi-server deployment. Please make sure that the load balancer will forward requests from the same client always to the same server (session stickiness).

2.5 Informatica BPM

The Informatica BPM "module" in the picture above is actually the full Informatica BPM product which is integrated using message queues or REST API calls to and from the Product 360 application server.

2.6 Physical File Share

All servers of the Informatica MDM - Product 360 deployment need access to the same physical file share in order to be able to distribute their workload inside the application server cluster. It is not necessarily a share, but all server nodes must have concurrent read and write access to the same folder structure in with normal file or file share access protocols. By whatever infrastructure this is provided.

2.7 Database

All application server are directly connected to the database. Multiple schemas are needed in the database. The database sizing should be done according to the specifications the database vendors provide. In general Product 360 is a OLTP application with a typical workload for these kinds of applications.

2.8 Elasticsearch

Fulltext search and Audit trail uses Elasticsearch server to store the data in indices. The Fulltext search features are integrated in the application server and directly communicate with the state of the art search engine from Elasticsearch. Index creation is done by the export and a specific post export step. The Audit trail module stores all changes anyone or anything does to the Product 360 objects in Elasticsearch. Audit trail records are written synchronously to the Elasticsearch Indices. Elasticsearch server is a mandatory component in case you want to use Fulltext search and Audit trail.

2.9 Message Queues

Message Queues are mandatory components of the Product 360 architecture. They are used for the integration between the Product 360 Server and BPM, Media Manager and also 3rd party systems.

3 Resource Limitations

A short overview on the resources the Product 360 application relies on.

3.1 CPU and CPU Cores

Most prominent resource these days are of course the CPU or CPU cores. The number of total cores in the application server more or less directly maps to the performance and maximum load capacity of the application. The Product 360 application server is optimized for multi-core systems in such way that mass operations are distributed in multiple parallel worker threads which are then processed by the available cores. In general this is true for all typical mass operations like import, merge, search & replace, data quality but also other operations in which a user can select multiple objects like classification, attribute maintenance and more.

So, the needed number of cores is bound to the number of concurrent users and jobs and how many objects those users and jobs are processing at the time. It's not really relevant how many objects are in the database in total.

3.2 Memory

The memory consumption, in contrary to the CPUs is not so much dependent on the number of users or jobs but on the total number of objects in the database. For large exports or lists a significant amount of memory is needed for the time the model is processed which makes it important to have enough memory resources available for those tasks. The number of concurrent users is not so important since the application server holds only very little user specific data.

3.2.1 Multi-Server considerations

Caches are not shared across multiple servers. So the memory needs are typical identical for all servers, no matter how many servers are used. Servers which serve web clients typically need a bit more memory as the client session consume also memory on the server.

3.2.2 Caches

Product 360 has various caches in place which store data in memory in order to reduce the number of requests to the database. For the memory requirements it's important to be aware that some of those caches are held completely in memory. In order to get a better impression we list a few of them here:

3.2.2.1 Status Cache

The status cache holds current results of all data quality checks for all objects which have such results in memory. This is needed in order to be able to filter lists of items very fast for these data quality results. The memory footprint is not as big as one would expect as those results are stored in a highly optimized form. Since the application relies on the availability of this cache and its completeness, you can not adjust the number of elements in this cache.

3.2.2.2 ListModel Cache

The Service API has the possibility to cache complete list models. This is useful for paged list access from Service API clients so they don't need to re-evaluate the query for every page. In case the Service API is not used correctly you might run into memory issues in case this cache is filled with really big list models. The maximum number of cached list model is configured in the `ehcache.xml` file. If you use V2 or later of the service api the list model cache is not used by default - it really only should be used in case of the paged list model access scenario.

3.2.2.3 ListProcessor Cache

For analysis purposes needed for example in the dashboards, we do have a ListProcessor cache. This cache holds all records for the requested fields in memory and is able to execute analysis logic on those fields. Mainly used by the dashboard its size is directly bound to the number of dashboards and how many fields are used in those.

3.2.2.4 Proxy Cache

The proxy cache loads the basic identifying fields of every entity item which is in the system during server startup into memory. So, for example, the cache has all identifiers of all products, variants or items and all other entities and also their internal database id's. In case an import or service api or other application modules now need to resolve an objects by their identifier, this cache is used. As the cache has all items in memory, he also knows what items do not exist in the database. So this cache not only improves performance for existing items but also for new items. Of course, millions of identifiers do have a memory toll - but the benefits are tremendous. This is why the cache is specifically calculated in the sizing calculator based on the number of items.

3.2.2.5 DetailModel Cache

Detail models might hold all data for a single object. It's in the nature of those "big" objects to need more time to load from the persistence storage. That's why the detail model cache comes into place. Multiple requests for the same objects will be handled by this cache. Every Desktop client has an own local version of the detail model cache which consumes all available and otherwise not needed memory. The server has a specified amount of memory for this and will not "just grab everything".

Every load of a detail model will put the data into this cache, every save or delete operation will invalidate the cache again. Typically objects like hierarchies, units, users and others do profit the most from this cache - as they are not changed that often.

3.2.2.6 Dashboard Component Data Cache

For specific dashboard components it is possible to cache the data, which is loaded from the data source and then presented to the dashboard user. This is particularly useful when the query (=Entity Report) behind the data source is very complex and time consuming.

A cache element within this cache equals to an entity item list of one dashboard component instance of one dashboard. A new cache element is also created if the already cached entity item list was loaded by a user with a different ACL combination than the current user, because then the content of the entity item list might differ.

For details on this cache and also the memory usage of it, please refer to chapter "Data caching for dashboard components" of the Configuration Manual.

3.3 Network

Product 360 is optimized for mass data handling of rich product data. Using powerful grouping functions in tables with millions of records in multiple views is a central key feature of the desktop client. The Desktop Client is designed and implemented as an intranet application which typically has at least 100 MBit network bandwidth available. Although the Desktop Client works also with less, you might experience longer waiting times in case you load larger tables. The actually needed bandwidth is very different depending on the actual use cases of the users so this sizing guide can not give a definitive answer to this.

Table views in the web and desktop client, the export and also the service api work with large table like structure with possible millions of rows for a few tens of fields. In order to provide these table models in a fast way the network bandwidth between application server and database server must be as best as possible. At least 1 GBit, better would be 10 GBit is required. It is also required, that the network connectivity between the Product 360 Servers is ensured constantly (network connection issues, that last longer than 1 minute will very likely end in a failover scenario and/or in unexpected behavior of the overall Product 360 system).



We do not recommend to have the database in a separate network or in the cloud. For cloud deployments you need to make sure that also the application servers are in the cloud and the network speed between application and database server is fast enough.

4 Scaling Factors

4.1 Users

When we talk about users there exist a lot of different definitions. For this sizing guide we will concentrate on two of them.

4.1.1 Licensed Users

The amount of individual users Product 360 has been licensed for. For the license it makes no difference if the user logs in via web or desktop client or via service api.

4.1.2 Concurrent users

To be able to size the system correctly, the purely licensed number of users is not really important. Some licensed users might work all day with the system, executing mass operations non stop while others just log in once in a while to check something. To be able to size the system correctly we need the number of concurrent users. Users which are logged in and work with the system at the same time, concurrently. This will typically be a mixture of power users and others - this has been taken into account in the sizing.

4.1.3 Concurrent Supplier Portal Users

External users which work with the supplier portal concurrently.

4.2 Product, Variants, Items

The number of products, variants and items in combination with their complexity is even more important for the database than it is for the application server in Product 360. Handling 100 million items is an easy task if those items just have a handful attributes. It's a different story in case the items have 200 attributes, multiple languages, hundreds of references to other objects and so on. Because of this all our benchmarks are performed with a defined set of objects, not only in number but also in complexity. This complexity profile is compiled based on the real world numbers of our customer base and therefore is a realistic profile.

4.3 Jobs

Background jobs like imports, merges and exports are at least as important as the users in terms of sizing. All jobs which modify objects are implemented to use multi-threading in order to utilize as much processing power as available on the machine. The more CPU cores, the faster they finish. Those background jobs are designed for mass data and perform best if executed for larger datasets.

4.3.1 Import

The import is divided into two phases.

4.3.1.1 Phase 1

- Reading the import file
- Combining multiple rows for the same object
- Executing any Import Functions.

Phase 1 is single threaded and takes usually 10% of the overall execution time, depending on the used functions of course.

4.3.1.2 Phase 2

- Executing the business logic to create or update the objects

Phase 2 is multi-threaded and uses all available CPU cores to perform this. The more cores, the faster it gets.

4.3.2 Merge

The merge has a similar load profile than Phase 2 of the import. It's multi-threaded and reads the supplier catalog item and performs business logic to create/update the master catalog item.

4.3.3 Export

The export uses the mass data table like structures to process huge item counts. Larger exports are executed multi-threaded to improve throughput. There is a dedicated thread pool for the export which can be adjusted. Please be aware that the export is quite memory intensive and each thread executing an export package can easily use gigabytes of data. Adjusting the size of this thread pool should only be done in case the server has enough memory for this task.

4.3.4 Data Quality

The data quality checks are usually executed automatically after an import, before a merge or before an export. It might also be scheduled regularly or manually through user interaction. The data quality job is also multi-threaded which means that the results are applied concurrently for all affected items.

As you can see, every job which creates or updates objects in Product 360 is implemented using multi-threading technology. If you have a lot of jobs running concurrently, it might be good to have one or multiple separate servers for job and data quality executions. Unless that, the jobs will be either slowed down by the users, or vice versa. *A single job will utilize only the CPU Cores of the server he is currently executed on.*

Since Version 10 data quality jobs that get triggered through an entity changed or entity created event are distributed on servers with the role MQ_CONSUMER_SERVER. The processing of all other data quality jobs did not change.

4.4 3rd Party

This category combines all 3rd party integrations which do not use import or export jobs for the integration. Typically 3rd party applications use the Product 360 Service API to read and write object data, trigger jobs or query the meta data. In case a 3rd party application sends a bigger list of objects which need to be modified, the Service API will also utilize multi-threading technology to perform this task as fast as possible. Because of this we usually recommend to have the Service API clients also use the job servers so they do not interfere with the Web or Desktop clients.

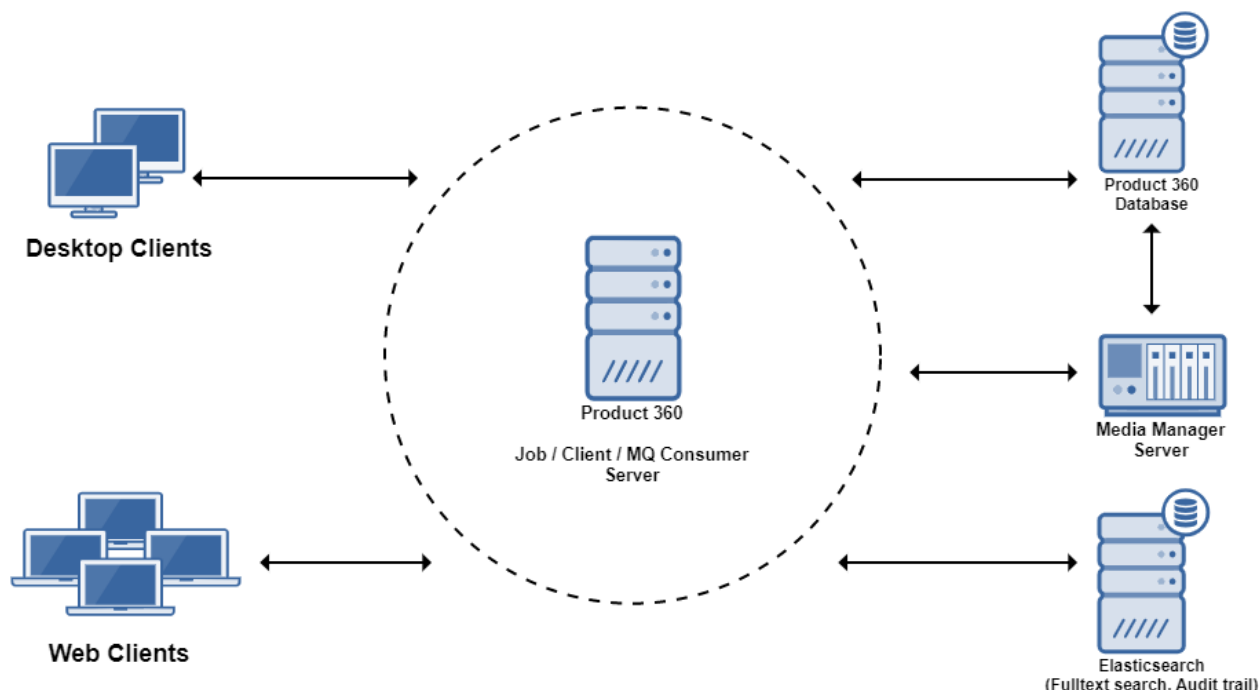
4.5 Workflows (BPM)

Workflows can be a huge factor when sizing a system. Workflows can be implemented in two fundamentally different ways. A workflow instance for a larger list of items, for example all items which have been imported or merger or going to be exported. On the other hand you can implement a workflow on a single item base which is triggered for item creation or modification. Especially in case of single item workflows the way they are implemented is really important. Workflows should not have long running synchronous Rest API calls to the application server. This would just bind the workflow thread too long and can lead to performance problems within the BPM engine. The best way to interact with the application server is by using the message queue api. Incoming messages on the message queue will be processed on all application servers with the MQ_CONSUMER_SERVER role. Internal batching mechanisms are used to efficiently process single item messages in batches.

To summarize this: Best workflows are always workflows which handle lists of items and not a single item per workflow instance. If the later is required use message queues to communicate with the application server. This increases the scalability and stability of the overall system. Calculated sizings always assume that single item workflows use message queue communication!

5 Logical Deployment

5.1 Single Server

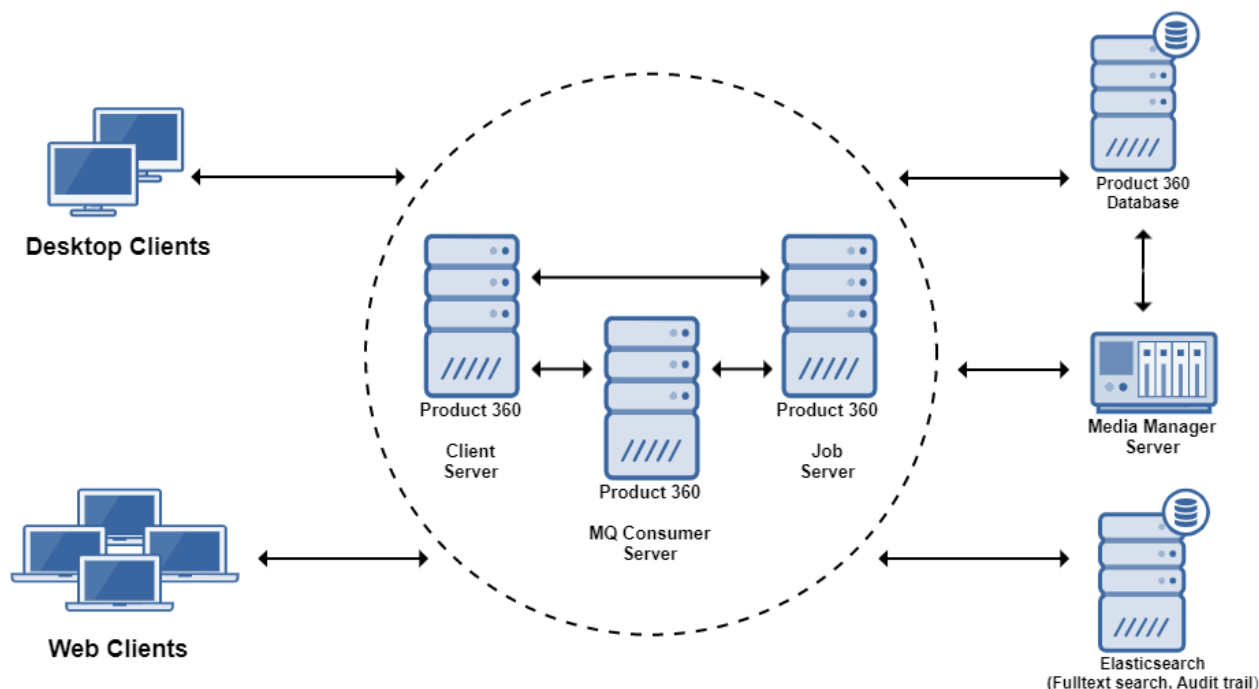


The database, the media manager and supplier portal are installed on different hosts. However, the core application server which processes all client requests as well as the jobs is deployed also as single server.

For small projects the single server is a valid option. It scales vertically with the number of cores. Also for scenarios in which jobs are only executed during non office business hours the single server deployment might be the right one.

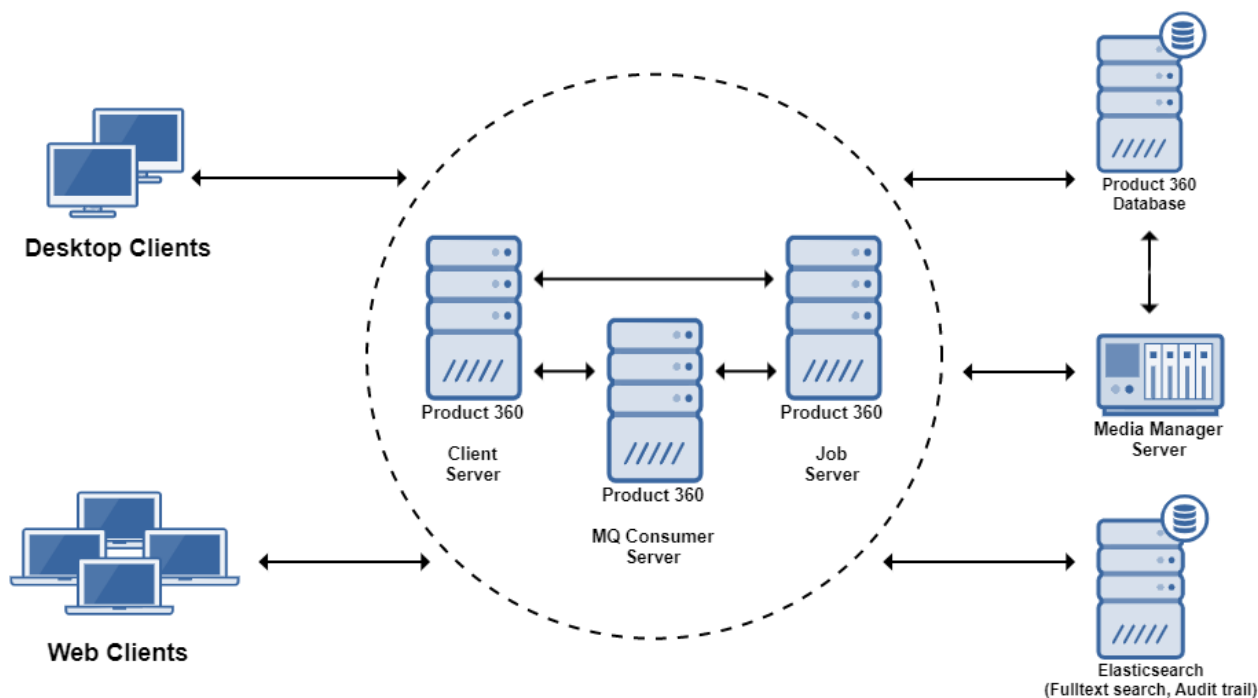
Note: For demo and development purposes it's also possible to install everything on a single host. We would only recommend this for productive use in very small installations.

5.2 Separate Jobs from Clients



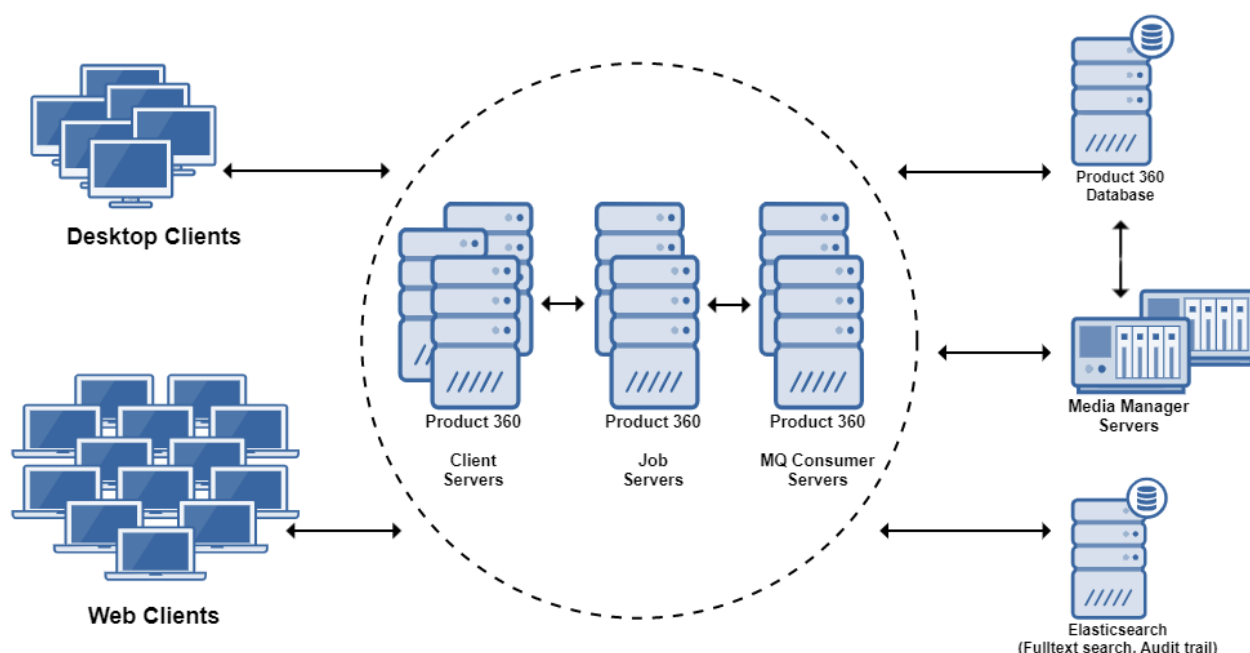
Depending on the number and type of background jobs and the time they need to be executed, a multi-server scenario is recommended. By separating the users from the jobs we make sure that response times for users don't degenerate, and job execution times also don't suffer from user load. Please note that the sizing calculator might return a relatively small number of cores for the jobs server - but usually it makes no sense to have less than 8 cores - or the jobs will finish not fast enough from a single job perspective. However, it's totally feasible to route also the 3rd Party applications to the job server as well as BPM calls so that the job server has a fair share of work to do also when he's not under full power by the job execution alone.

5.3 Separate Servers by Role



A multi-server scenario with a server for every server role is recommended depending on the number, type and execution time of background jobs, user load and incoming messages from message queues. By separating the different roles we make sure that execution times and response times don't suffer or degenerate for the other servers if one is on high usage.

5.4 Scale Out



Based on the expected load every module and server can be scaled out horizontally. You can have multiple user and multiple job servers, as well as multiple media manager or supplier portal servers. The physical deployment section will show in detail how this can look like.

6 Physical Deployment

6.1 Roles

PIM servers are deployed absolutely identical. Thus, in theory, all of them can perform all operations. To be able to separate the type of load, we introduced server roles. Of course it is absolutely possible that a server has multiple, if not all roles.

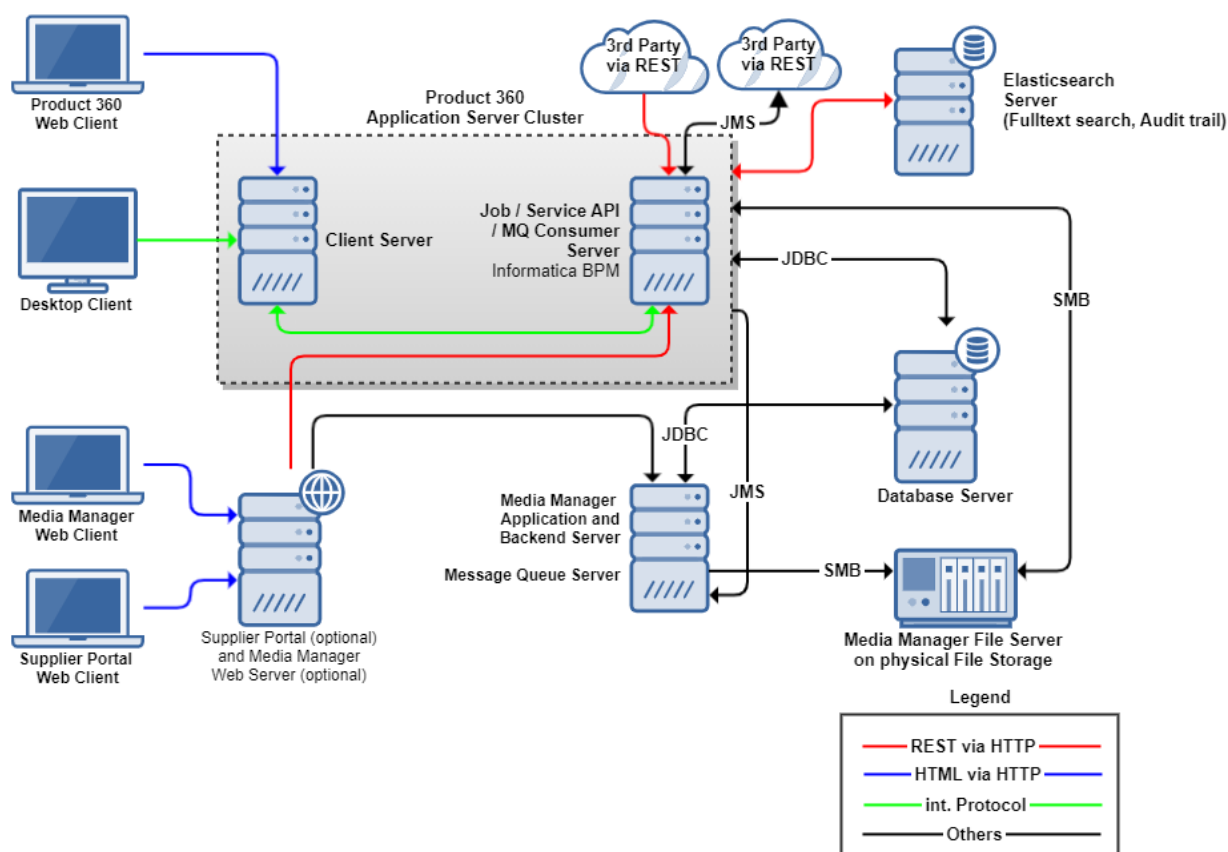
- **CLIENT_SERVER**
The server is able to serve desktop clients. Desktop clients will be distributed across all servers which have this role
- **JOB_SERVER**
The server is able to execute background jobs. This includes user triggered jobs as well as scheduled jobs. So pretty much all jobs which can also be seen in the process overview. In case multiple servers have this role, the jobs will be distributed across them using a simple round robin algorithm.
- **MQ_CONSUMER_SERVER**
The server is able to read and consume messages from a message queue. In case multiple servers have this role, the messages will be distributed across them.
- **PRIORITY_JOB_SERVER**
The server is able to execute priority jobs. Which job types are priority jobs can be configured in the

plugin_customization.ini configuration file. In case a priority job server is active in the network, priority jobs will only be processed by him. In case there is no priority job server started, those jobs will be executed by the "normal" job servers. In case there are multiple priority job servers, the normal load balancing mechanism will be used to balance the jobs across all priority job servers. (This role has been introduced with Hotfix 1)

Besides the Server roles which are configured in the server, the web/rest load balancing allows us to define two additional roles which need to be configured in the load balancer.

- Web Clients
This server only serves Product 360 Web Clients.
- Rest Clients
This server only serves Service API, or, REST clients (including the Supplier Portal)

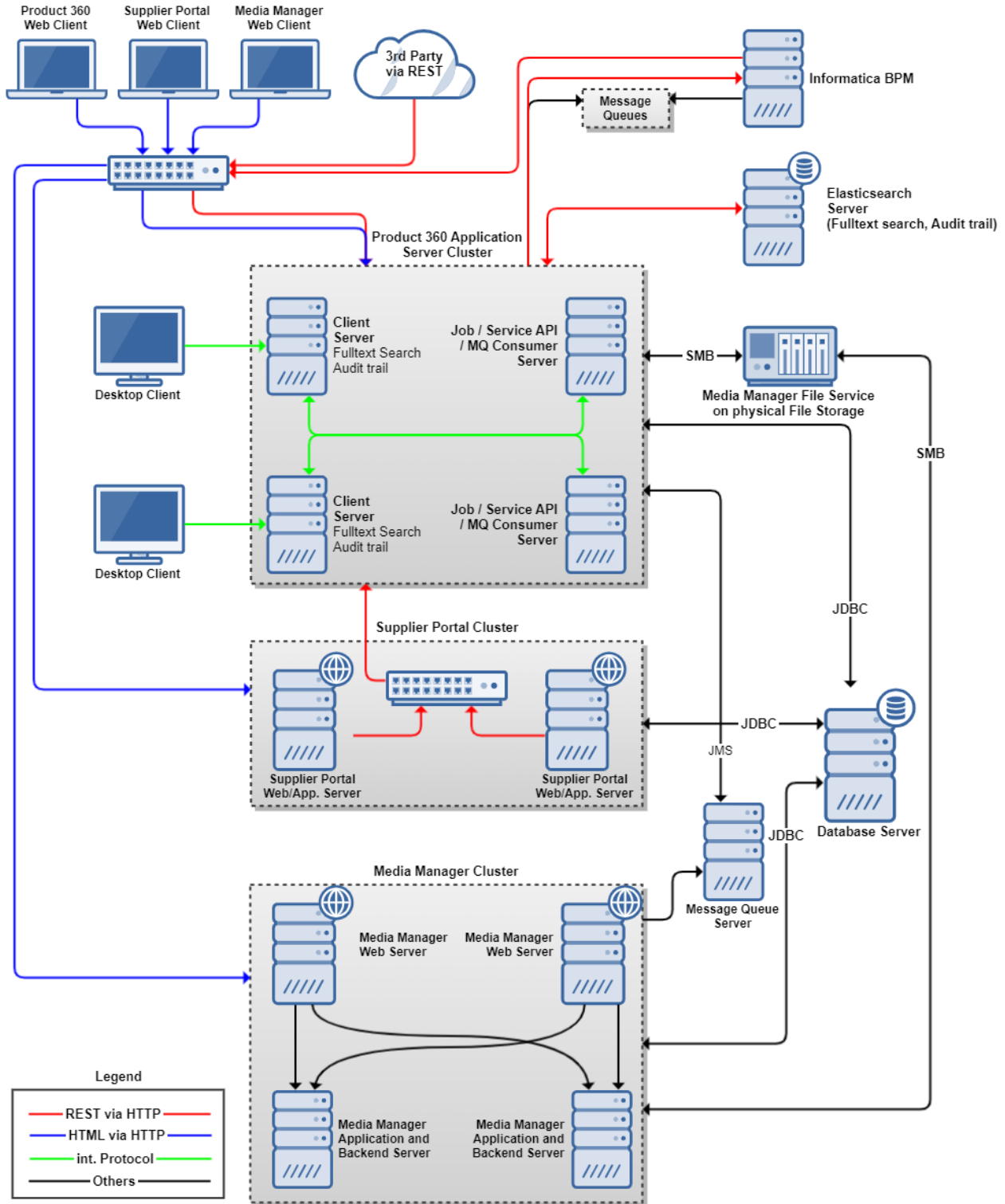
6.2 Typical Deployment



The typical deployment corresponds with the "Separate Jobs from Clients" scenario from above. In this picture you can see the communication paths from and to the modules of Product 360. Since this deployment does not have multiple servers for the web or rest clients, we don't need an HTTP load balancer. Optional modules like the Supplier Portal can be combined on a single host in order to minimize the number of hosts which need to be managed. Used Communication protocols include REST via HTTP, SMB for file transfer on the server side, as well as our own internal high-speed protocol for the communication between the servers and the desktop clients.

6.3 High Availability with Load Balancing

The so called "Full Blown Deployment" shows a scale out of every Product 360 module. It includes HTTP Load Balancer technology for all HTTP based clients (incl. REST). Modules are separated as much as feasible and scaled out individually. Please see this deployment as an example to what is possible with Product 360 Version 8 - not as a deployment which most users need.



6.3.1 Application Server with User Role

6.3.1.1 Load Balancing

Desktop clients automatically distribute them self on the available servers with the "CLIENT_SERVER" role. The used algorithm is based on the "least number of connected clients". At the very first startup of the client, a valid application server address is required. This can be just any server from the network, no matter which roles are configured. From this server, the client receives the hosts for all available client servers and automatically connects to the one with the least amount of connected clients.

6.3.1.2 Fail-over

In case the very last server with the "CLIENT_SERVER" role crashes, *all* other servers will automatically accept connection requests from clients. This build-in fail over logic is triggered automatically when the client reconnects to the network. Since the client remembers all servers of the network, he will try to connect to the very first one which is responsive to determine the current list of available servers to choose from.

6.3.2 Application Server with Job Role

6.3.2.1 Load Balancing

All servers with the "JOB_SERVER" role have a fully functional job framework up and running. Whenever a new job is scheduled the job information is written to the database. All Job servers are polling the job tables in regular intervals to see if new jobs are available which should be executed. The first server with a free Job Execution thread will pick the new job and executes it. It's not possible to define in advance that a specific server should execute a specific job. The distribution algorithm is round robin within the job servers. Internal system critical jobs are preferably executed on job servers.

6.3.2.2 Fail over

In case a Job server fails, all jobs he currently is executing are failed also and will be shown as cancelled. Users need to reschedule those jobs again manually in case it's not a regular job. For regular jobs, the next execution will be triggered as soon as it's time for it. A crash of the very last job server will trigger a startup of the internal system critical jobs on the client servers. User jobs can still be scheduled, but will not be executed until a job server is running again.

6.3.3 Application Server with MQ Consumer Role

6.3.3.1 Load Balancing

Messages from queues automatically distribute themselves on the available servers with the "MQ_CONSUMER_SERVER" role to be consumed.

6.3.3.2 Fail over

In case one server with the "MQ_CONSUMER_SERVER" crashes the remaining and new incoming queue messages will be consumed by the other server with the specified role. When the crashed server was currently consuming a message this message will be put back on the queue so another server can consume it.

In case the very last server with the "MQ_CONSUMER_SERVER" role crashes, the remaining and new incoming queue messages will not be processed and stay on the queues. As soon as a server with the "MQ_CONSUMER_SERVER" role is started the messages will be processed again.

6.3.4 Application Server with Web or Service API Clients

For Web or Service API clients no special role configuration is needed on server side. Which server of the cluster serves Web Clients and which Service API clients must be configured within the Load Balancer. All servers of the network are, in general, capable to serve web as well as rest clients. Please see the Informatica MDM - Product 360 Configuration Manual for a configuration example.

6.3.4.1 Load Balancing

Web and Service API clients should be balanced using a standard HTTP Soft- or Hardware Load Balancer. We recommend to use the Apache Load Balancer but in general customers can use any state of the art Load Balancer which supports sticky sessions. The balancing algorithm to use can be decided by the customer depending on the capabilities of the Load Balancer.

6.3.4.2 Fail-over

In case the application server for web or rest clients crashes the clients need to reconnect, or re-execute the REST request which failed during execution. The HTTP Load Balancer will automatically direct the client to a still available server. It's recommended to configure the Load Balancer so it has a fall back server which is used only in case the primary servers are not responsive any more.

6.3.5 Media Manager File Server

6.3.5.1 Load Balancing / Fail over

The Media Manager File Server is a share of a Windows or Linux based server system. Please refer to an OS specific cluster solution. To prevent data loss use a professional backup solution or use at least a raid system for your shares.

6.3.6 Media Manager Web Server

6.3.6.1 Load Balancing

The Media Manager Web should be balanced using a standard HTTP soft- or hardware Load Balancer. We recommend to use the Apache Load Balancer but in general customers can use any state of the art Load Balancer which supports sticky sessions. The balancing algorithm to use can be decided by the customer depending on the capabilities of the Load Balancer.

6.3.6.2 Fail-over

In case the Media Manager web server crashes the clients need to reconnect. The HTTP Load Balancer will automatically direct the client to a still available server. It's recommended to configure the Load Balancer so it has a fall back server which is used only in case the primary servers are not responsive any more.

6.3.7 Media Manager Server

Load Balancing

Use horizontal installations (one Media Manager Server installation per machine) of the server modules to ensure a high work speed while increasing of the connections or jobs.

Fail over

Use horizontal installations (one Media Manager Server installation per machine) of the server modules. If one server is no longer available the other server(s) will take over the jobs.

6.3.8 Database Server

Please refer to the deployment manuals for the Oracle or MS-SQL Server databases for high availability and fail-over deployment strategies.

6.3.9 Supplier Portal Server

6.3.9.1 Load Balancing

The Supplier Portal is a web application and should be balanced using a standard HTTP soft- or hardware Load Balancer. We recommend to use the Apache Load Balancer but in general customers can use any state of the art Load Balancer which supports sticky sessions. The balancing algorithm to use can be decided by the customer depending on the capabilities of the Load Balancer.

6.3.9.2 Fail-over

In case the Supplier Portal server crashes the clients need to reconnect. The HTTP Load Balancer will automatically direct the client to a still available server. It's recommended to configure the Load Balancer so it has a fall back server which is used only in case the primary servers are not responsive any more.

6.3.9.3 Configuration

Most of the configuration options are shared between all Supplier Portal Server instances. However, there are some things to consider:

- All instances must have access to the same directory share that is used for file storage
- Background jobs for import job synchronization should be configured to run on one server only

Depending on the expected load, all internal http communication can be either routed to a single Product360 Application Server or load-balanced to multiple Application Servers. The same applies for the Media Manager Server.

For details, check the Supplier Portal Configuration Guide.

6.3.10 Informatica BPM Server

Please refer to the Informatica BPM server documentation for details on how to set up a multi-server and high availability development.

6.3.10.1 Fail-over

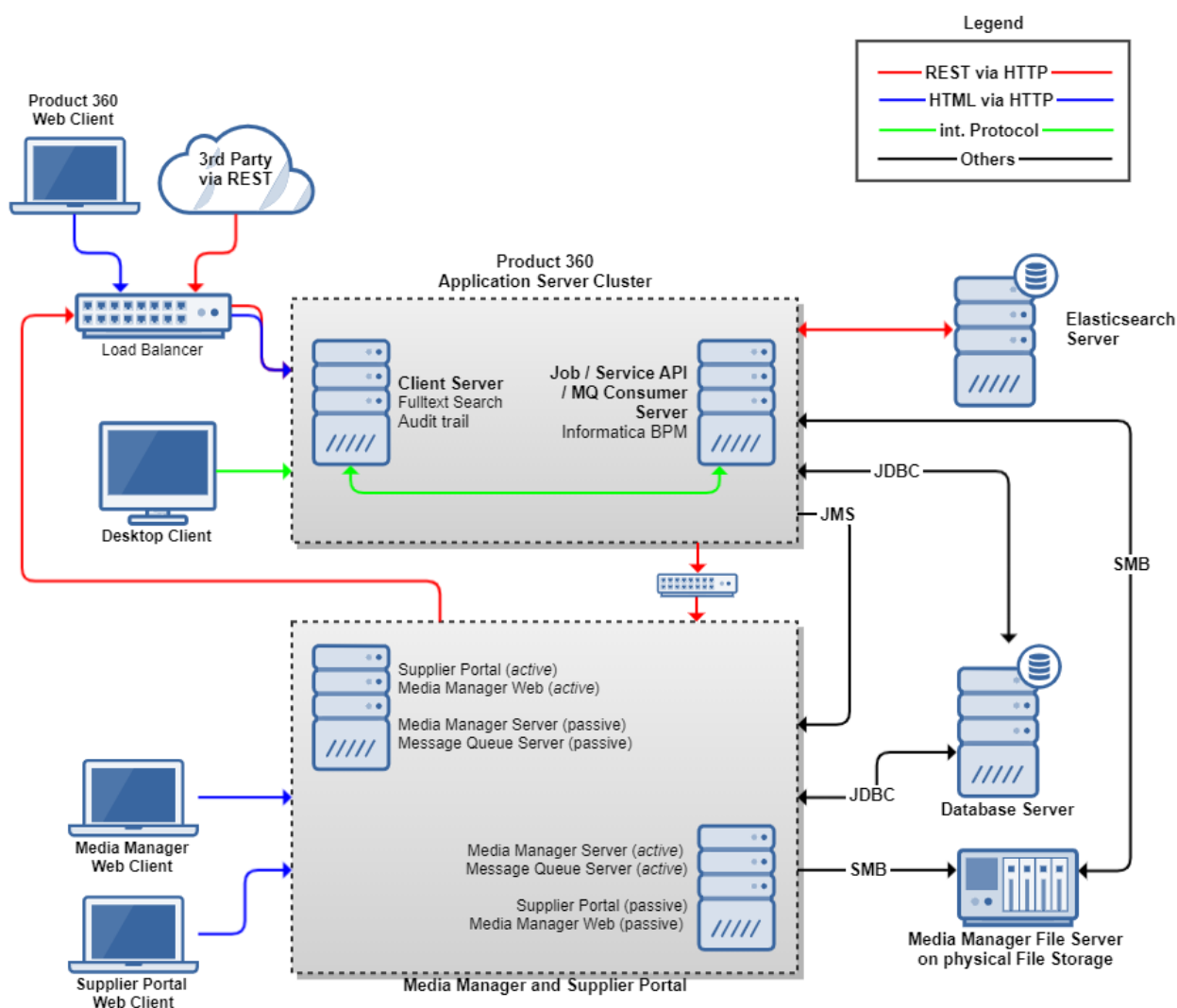
The MDM - Product 360 Application server will buffer all requests to the Workflow engine in case the BPM server is not available. So clients can keep up working and the workflow requests will be processed as soon as the BPM server is back online.

6.3.11 Message Queue Server

By default Informatica MDM - Product 360 uses the Apache Active MQ which can also be deployed in a high availability mode. Please refer to the available documentation for Apache ActiveMQ or any other 3rd party message queue system.

6.4 High Availability without Load Balancing

As described earlier, the Informatica MDM - Product 360 deployment has a maximum of flexibility. The following is an example of how you could reach a highly available system without the need for laying out each module as a multi-server environment. Of course, the load scenarios for this kind of deployment are limited, but it might be sufficient for a large group of customers.



6.4.1 Application Server Cluster

Since the Informatica MDM - Product 360 Application server has a build-in fail-over logic (as described above), it's a good practice to just have two server instances. One for clients, one for jobs. If either of those crashes, the other one will take over his responsibility in the meantime. To be close to transparent for the clients, you would need to have a Load Balancer for this cluster - so all web client requests and service API requests are automatically routed to the server which is still working. The routing for the desktop clients is build in already. It's important that **all** HTTP traffic is routed through the Load Balancer otherwise the interconnection between the modules might not work.

6.4.2 Media Manager and Supplier Portal

The services for Media Manager Web, Media Manager Server and Supplier Portal are installed on two hosts. Each service is part of an OS cluster configuration in which some of the services are active on one host, and passive on the other. In case of a failure of either of the hosts, all services which are active on the failed one will be moved to the other host. So during a failure the services are still up and running. Depending on the

available hardware they might be slower since the host needs to handle now all services, but they are still available. For the sizing of the hosts it must be considered that each host needs at least as much memory as all services together require in their minimum specification.

6.4.3 Media Manager File Server

The Media Manager File Server is a share of a Windows or Linux based server system. Please refer to an OS specific cluster solution. To prevent data loss use a professional backup solution or use at least a raid system for your shares.

6.4.4 Database Server

Please refer to the deployment manuals for the Oracle or MS-SQL Server databases for high availability and fail-over deployment strategies.

7 Limitations

7.1 Number of Servers in the P360 Server cluster

A P360 Server cluster can in theory hold a unlimited number of servers. In theory means that currently there are no build-in limitations on the number of servers - but the synchronization and communication overhead will increase with the number of servers. Please contact Informatica Support if you plan to use more than 4 application servers (not counting the servers of the other clusters like Media Manager!).

7.2 Database Performance

As you can see in the diagrams, the database seems to be the next logical resource bottleneck. As of today this can only be solved with vertical scalability on the database. Please contact your DBA for details on how to achieve the needed performance on the database.

7.3 Elasticsearch Performance

Elasticsearch can have performance bottleneck based on data volume, usage and data retention time for Fulltext search and Audit trail. To achieve better performance, have separate clusters of Elasticsearch for both Fulltext search and Audit trail.



We do not guarantee that sizing parameters fit on virtual hardware. Our sizing recommendations are developed and measured on non virtual machines. Please note that performance on virtual environments can differ and is therefor not guaranteed. However, we still recommend to use current

state of technology virtualization environments to be able to grow the servers also vertically. It's just important that the CPU cores from the sizing are really available in a physical way and not shared with other applications. A close monitoring on the host hardware is needed to inspect and adapt the needed resources to the everyday needs of the customer.

Copyright

© Copyright Informatica LLC 1993, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.