Purpose

The purpose of this document is to describe the MobileFrame Server and its components in depth and to provide suggestions for setting up your server(s) to maximize scalability and performance in just about any situation.

Audience

This document assumes you have an understanding of the MobileFrame back office architecture and a familiarity with the MobileFrame terminology and components. This document is geared to IT professionals or users needing more information on how to set up the MobileFrame Server components for their requirements.

So… "How do we scale?"

The most frequent questions we're asked in engineering from prospective customers are "how do we scale?" or "how many servers will I need?" These are impossible questions to answer since every set of requirements, each application, each network, each corporation's security policies and infrastructure, and the usage patterns are different and out of the control of MobileFrame Engineering. It's therefore not possible for anybody to tell you with perfect accuracy how to set up your MobileFrame Server(s) or how many you'll want to have to meet your requirements before hand. Best guesses are all that can be offered.

The better question to ask is "Can you scale?" and the answer to that question is yes. The MobileFrame architecture, and in particular the MobileFrame Server, has been built for the enterprise. So although the entire MobileFrame Server can be run on a single machine if you want, you can slice and dice the components any way you see fit to get the performance and scalability that you need.

The MobileFrame Server

The MobileFrame Server is actually a set of components that work together and run together to perform the back office operations of the system. Here's a brief summary of each component and what its job is within the context of the MobileFrame system.

- **The MobileFrame Database.** This database can reside on any supported DBMS (SQL Server, Oracle et al). It contains all meta data describing user objects, the system tables for the MobileFrame operations (tasks, projects, users etc), the logs used throughout the system, the results captured in the field, and the tables that manage synchronization. All software components operate on this database.

- **The Sync Server.** This software component resides within IIS as part of the MobileFrame Server components. Its job is to listen for requests from clients in the field and fulfill those requests. For example, it sends new projects, new object definitions, new enterprise data, and new tasks to the clients that are supposed to get them. It also receives results from the field, and performs several housekeeping functions like authentication, setting client preferences, and sending system updates.

- **The Database Monitor.** This software component resides within IIS as part of the MobileFrame Server components. Its job is to maintain the integrity of the information in the database. For example, it looks at enterprise data that has been synchronized and resynchronizes those elements if they have changed in some way or if the relationships have changed in some way. It also performs functions like activating or deactivating a project based on the start and end time settings.

- **The Workflow Server.** This software component resides within IIS as part of the MobileFrame Server components. Its job is to run server-based workflow, that is, workflow on any task that is to fire when results have been received on the server.

Each of the software components in the server have their own settings for tweaking the efficiency of each component or the enabled feature set of the components. A description of each of these settings is beyond the scope of this document as they are already outlined in detail in the administrators guide and via the support site.

One setting that all of the software components have in common, however, is the ability to deactivate the component for that server. This is important because by giving the user the ability to configure which components are running on which server; they can create any necessary permutation of components to fulfill your back office requirements.

A Word About Licensing

The MobileFrame Server is a combination of components. They can all run on one machine or run on multiple machines. You can choose to disable or enable the components of the server as needed. Please note that the MobileFrame Server is licensed by the machine (not CPU). Depending on which MobileFrame Suite you purchased, you may have up to 4 machine licenses built in to your license. At any time you can add additional machine licenses, but the default number of licenses is listed below. These are accurate as of Sept 2005.

MobileFrame Standard Suite	1 Machine License included.
MobileFrame Business Suite	2 Machine Licenses included.
MobileFrame Enterprise Suite	4 Machine Licenses included.

Check with your sales person regarding the purchase of additional machine licenses should you decide you want or need them.

Load Balancing and Parallel Processing

Because you can repeat components on other machines and have multiple instances of the same component running against the same database, you can distribute the work of the components across multiple machines should you need to in order to satisfy your requirements. For example, if your server hardware isn't able to keep up with the demand of all your users synchronizing, you could have two servers running the Synchronization Server components and use a software or hardware load balancer to spread the traffic between them. You could add more servers if you're still finding that to be inadequate.

In addition, some of the components will load balance automatically if more than one is running. The Workflow Server, for example, can be repeated across multiple machines if you're finding that you can't keep up with the processing of workflow on results coming in from the field.

Scalability Suggestions/Features of the Synchronization Server Component

The synchronization server component is the most used component. It's where the clients connect and request/send information. To minimize database access, the synchronization server will cache database connections and other frequently accessed data. The cache settings and batch settings can be configured on the server component so that you can find the "sweet spot" for your implementation. For example, if your users are generally synchronizing over a stable connection (cradled or reliable Wi-Fi), you can increase the batch size to get more information per synch. However, if your users are synching in the field over a connection that may get severed frequently (cell phone data connections for example), you may want to lower your batch size. There are Knowledge Base articles on these settings.

If you're finding that the traffic is too much for a single synchronization server to handle or the response time to sync is higher than you want, you can put a second synchronization server on the network. You can then use a hardware or software load balancer to automatically route traffic between the various machines.

Another alternative would be to manually load balance the sync servers by giving separate URLs to different users. For example, assume you have 3000 users and you're finding that one sync server isn't responding adequately when too many users are synching at one. You could set up another sync server on a different URL and give 1500 of those users the other URL. Your sync servers could be broken out by department, territory, time zone, or anything that makes sense to your needs.

Alternatively, you can reduce what is being synched in the application. Verify that the steps in the task that aren't needed on the server are marked to not sync back. Also verify that the project doesn't contain data that isn't being used on the server.

Scalability Suggestions/Features of the Database Monitor Component

The database monitor doesn't really suffer much from "too many users" since it is a relatively invisible component that just maintains integrity behind the scenes. However, it can use memory and CPU power when it's found a lot of things to do.

If your synchronization server is low of free memory or CPU cycles, consider putting the database monitor on a different machine so it can do its job without fighting for the CPU while synchronization is going on. Other options on the database monitor, such as frequency of looking for changes, can affect how active the database monitor is.

Currently the database monitor does not provide features to break its processing to just specific objects on the database, although this is planned.

Because the database monitor needs access to the database but isn't used by the clients, it can easily be broken out and put behind a firewall if your synchronization server is exposed publicly (for use over the Internet).

Scalability Suggestions/Features of the Workflow Server Component

The Workflow Server can be completely unused or the most used part of the system. If your tasks don't have any server workflow, for example, then this component will do nothing and can just be disabled.

If your results go through extensive server processing though, particularly if part of the workflow accesses external systems that you will need to wait on, you will want to consider breaking the workflow server component onto a separate machine or multiple machines.

One interesting feature of the Workflow Server is its ability to automatically load balance. As results come in via the sync server, the sync server puts into a "queue" a request to process workflow for that set of results. The Workflow Server(s) works off this queue by pulling a request off the top, processing it, and discarding it. If you have multiple Workflow Servers running simultaneously, they'll share the queue. So if you have a workflow operation that takes 1 minute to complete, you can process one set of results a minute with a single server. If you put 3 more Workflow Server's in your system, you can now process 4 a minute.

If you're finding that your workflow is getting backlogged, it's probably because your results are coming in faster than your workflow server(s) can keep up with the processing. In these scenarios, you'll definitely want to add additional workflow servers to your system.

The Workflow Server can be broken out and put behind the firewall should you want to process workflow against systems that are internal to the organization but not exposed to the Internet.

Mix and Match – Typical Configurations

Because of the flexibility of the components and how they're configured you can mix and match your functionality any way you want. Here are some typical configurations to get your started:

The Standalone System

This system is ideal for smaller implementations in which the number of simultaneous connections will be well within the limits of the hardware. In this configuration, the MobileFrame Server components are all on one machine. This includes the database, sync server, database monitor, and workflow server components. It's easy to set up and requires the least amount of hardware.

The Business System

This system is suited to smaller to mid-size implementations where a back office integration is usually desired. In this configuration, the database, database monitor, and workflow server are on one machine, sometimes behind a firewall and the sync server is on a separate machine. As the business grows the system can be broken out onto other machines or multiple instance of the same components can be enabled.

The Enterprise System

This system is completely broken out. All components reside on their own machines. In general, this level of scalability won't be required except in the most rigorous of environments. Multiple sync servers are load balanced in front of the firewall. Multiple workflow servers and/or database monitors are behind the firewall. Sometimes there is still just one database, but in very extreme cases, you could have two databases as well; one for managing tasks/projects and another one for holding results and processing workflow.

Planning and Implementing

So with all this information, what do you do next? As was mentioned earlier, there is no way to do anything but guess up front with what you'll need so plan for this ambiguity by defining your requirements up front and planning a phased-in implementation.

Defining your requirements up front

By knowing what your limitations are, you can have a better handle on how much room you have to guess in. Here are some questions you should have answered during your planning phase:

- *What is an acceptable response time?* Keep in mind that the MobileFrame clients synchronize in the background so it's not like the users are having to wait around for sync to do their work. A customer that requires sync to finish in 10 seconds will have different requirements than one who is willing to wait up to a minute.

- *What is an expected usage pattern?* If you have an implementation of 1000 users and they all sync at different times throughout the day, the total number of concurrent users at any given time is relatively low- thus the server requirements will be low. However, if the system has been set up to sync them all at 8:00am, you'll need to have a server system capable of handling a burst of 1000 users at the same time. Designing the tasks or scheduling the sync times for the user so they are more evenly distributed is the best way to not have to accommodate "bursts".

- *What is the server workflow like?* If you have a simple and quick to process workflow you can keep up with demand easily. However, if your workflow is accessing external systems that can take minutes to respond, your workflow queue will get backed up quickly. The workflow server will continue to progress the queue even during the evening, but if there is too much it may not be able to keep up. Pay attention to the server workflow requirements and add more workflow servers as necessary.

- *Are there any "data transfer" jobs that affect the system?* If you have jobs to shuttle information between MobileFrame and some other system, don't schedule them during the day when the system is busy. Also, make sure you give time for the database monitor to realize the new changes before the next day.

Implementing

It is highly recommended that you phase in your implementation. Put a small amount of users on the system first and monitor to see if your expectations during the planning phase are actually accurate. You may find out, for example, that users are synching half as often as you expected or that the server is handling things way more efficiently that you thought. Or conversely, you may find that your workflow is taking longer to run than you thought.

By planning up front you can get a best guess starting point. By phasing in your implementation, you can adjust your starting point before things get out of hand. If you're finding that you can barely process your workflow with 50 users online, and you're going to roll out to 200, you can either adjust the workflow to go faster, add new workflow servers, or improve the hardware before adding the remaining users. What you don't want is to put all 200 users on the system on day 1 and find out your workflow server is choking. Because you'll have a much harder time finding out at what point it was becoming unacceptable and will have wasted those users' time.

Where To Find More Information

The information in this document should help give a better understanding of what will be involved in setting up your back office to most effective. In addition to the suggestions in this document, it is recommended that you check our documents on our scalability tests to get a better idea of what to expect and on what hardware to expect it. In addition, there are documents describing the various options of the MobileFrame Server components in more detail. All of this information can be found in our Knowledge Base or can be requested from your sales representative.