

InterWorx Server Administrator MySQL Guide

by InterWorx LLC

Contents

I	MySQL administration via NodeWorx	3
1	An Introduction to InterWorx's MySQL Integration	4
1.1	Compatibility With InterWorx	4
1.2	Additional abilities	4
2	NodeWorx MySQL Overview Page	6
2.1	Detailed descriptions of Settings found on Overview Screen	6
2.2	Additional features of Overview Page	10
3	Remote Servers	13
3.1	The Remote Servers Screen	13
3.2	Adding a remote server	13
3.3	Setting a default MySQL server	14
3.4	Deleting a remote server	14
3.5	Managing/Configuring a remote server	15
4	phpMyAdmin	16
4.1	Basics	16
4.2	Common Session Issue	16
II	Advanced MySQL Administration and Management	18
5	Tracking down load issues	19
5.1	Installing mytop	19
5.2	Using mytop and logging into the MySQL database as root	20
5.3	mytop Display (From Official Documentation)	20
5.4	General Paradigm for Tracking Load Issues	22
6	Data backup, repair, recovery	23
6.1	Backup inside InterWorx	23
6.2	Backup on Command Line	29
6.3	Checking and Repairing corrupted data	31
7	Migrating a SiteWorx account to a different MySQL server	33
8	Odds and Ends	34
8.1	Command-Line Reference	34
8.2	How MySQL users and databases are mapped to SiteWorx accounts	35
8.3	Clustering Considerations	35

Preface

Databases are the fundamental data storage mechanism for web applications today on the internet. The InterWorx hosting control panel supports the web's most ubiquitous database server, MySQL. It's an open source SQL relational database server currently maintained by Oracle Corporation. Due to it's wide deployment and integration, most users feel comfortable using the MySQL database system. In addition, the online documentation for the software is very robust, making it easier than ever to become an expert in the database system.

In order to get the most out of this guide, we recommend that a server administrator have some experience with the following:

- Linux command-line
- Starting/Stopping daemons using `/etc/init.d/servicename` or the “service” command
- Basic SQL queries (SELECT, INSERT, UPDATE, DELETE)
- phpMyAdmin
- MySQL server administration knowledge (`/etc/my.cnf`, `/var/lib/mysql/*`, mysql CLI client, mysqldump, mysqlcheck)

If you don't have experience with one of the above items, you can still probably get use out of this guide, albeit with perhaps occasionally relying on Google to look up unknown terms or proper use of a piece of software. We try to leave most of the explaining of how MySQL works to the MySQL documentation and instead try to provide documentation here that is useful to an InterWorx server administrator.

Part I

MySQL administration via NodeWorx

Chapter 1

An Introduction to InterWorx's MySQL Integration

InterWorx interfaces with the MySQL server software much like you'd expect. For example, configuring options interacts with the server's main configuration file `/etc/my.cnf`. Additionally, in order to determine whether the service is online, the `/etc/init.d` service control script is used to query the status of the service. The user-friendly web interface can be located inside NodeWorx under System Services > MySQL Server.

1.1 Compatibility With InterWorx

InterWorx does not provide the RPM package for the `mysql` server. Instead, we rely on the operating system's maintainers to build and distribute the MySQL package for their system. It is always possible to upgrade or switch sources of your MySQL server as long as the following requirements are met:

- The MySQL server version must be version 4.0 or higher.
- In the `/etc/my.cnf`, the `old-passwords` setting must be turned on (set to `true`). This is because the MySQL client that InterWorx uses to interact with the system MySQL server is a 4.0 client, which allows for compatibility across multiple MySQL server versions.
- On InterWorx install, InterWorx should be able to access the root user of the MySQL without a password. This is the default for most Red Hat based fresh OS installs.

1.2 Additional abilities

1.2.1 phpMyAdmin

InterWorx has a fully integrated version of phpMyAdmin which allows SiteWorx users and NodeWorx resellers to manage all MySQL databases under their account, even if the databases are segregated across multiple MySQL users. In addition, from NodeWorx, administrators are able to see all databases for all users on their servers and manage them via phpMyAdmin. Our version of phpMyAdmin sits behind the authentication mechanism of InterWorx and thus is not accessible from the outside without providing correct NodeWorx or SiteWorx credentials (depending what panel you are logging in to).

1.2.2 Remote Servers

In the event you don't want to use the localhost MySQL server, InterWorx supports adding "remote" servers and assigning SiteWorx accounts to the remote servers. This allows you to segregate MySQL CPU and memory requirements on a separate server. This also allows you to scale the memory allotted to the server up for superior performance since a remote MySQL server can be set to do "only" MySQL. In addition, the remote server is not required to run

InterWorx or even an RHEL compatible operating system - you can choose what ever operating system you feel most comfortable administering a MySQL server on.

If you never want to use the localhost server, it is also possible to designate a remote server as the default MySQL server, and all SiteWorx accounts will use the remote MySQL server by default. This is covered in more depth in section [3.3](#).

Chapter 2

NodeWorx MySQL Overview Page

The overview page, located in NodeWorx under System Services▷MySQL Server▷Overview, allows your standard InterWorx Start/Stop/Restart of the MySQL server daemon. In addition, you are able to modify whether the service starts on boot¹. Auto-restart MySQL will cause InterWorx to meticulously monitor the mysql service and attempt to start it back up if the service remains down for a predetermined amount of time (this will prevent InterWorx from restarting the daemon in the event the service is already restarting automatically due to log rotation or manual restart). While these are pretty standard for all InterWorx services, what is not standard are the server-specific settings which are configured by modifying the `/etc/my.cnf` configuration live.

2.1 Detailed descriptions of Settings found on Overview Screen

Tweaking your MySQL configuration is fairly simple via the MySQL overview screen

2.1.1 Connections(max)

This determines the maximum number of simultaneous client connections. Keep in mind increasing this value also increases the maximum amount of UNIX file descriptors that the server needs. If that statement means nothing to you, just interpret it as “the maximum resources my server will need will increase”. You need more memory to manage all those connections, more CPU time to handle all the concurrent activity. On the other hand, if your MySQL server is not particularly greedy, you may be starving the applications which rely on the server for data storage. According to MySQL’s documentation, the default connection count is 151. Keep in mind that in a typically single page load for something like WordPress, a connection is held open for a fraction of a second. The value of your MySQL connection count should only be raised if you find that your applications are reporting errors of being unable to connect due to the max connection limit being hit (MySQL will tell you this). These errors will typically manifest themselves in the error logs of the SiteWorx account with the application installed.

2.1.2 Connection Errors (max)

This sets the `/etc/my.cnf` value `max_connect_errors`. The MySQL documentation says the following:

If more than this many successive connection requests from a host are interrupted without a successful connection, the server blocks that host from further connections. You can unblock blocked hosts by flushing the host cache. To do so, issue a `FLUSH HOSTS` statement or execute a `mysqladmin flush-hosts` command. If a connection is established successfully within fewer than `max_connect_errors` attempts after a previous connection was interrupted, the error count for the host is cleared to zero. However, once a host is blocked, flushing the host cache is the only way to unblock it.²

¹This is done on the back-end by setting the service to “on” via `chkconfig`

²http://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html#sysvar_max_connect_errors

MySQL Server Control [?]

Version: [?] 5.0.95

Status: **RUNNING**

Action:

Start on boot-up: [?] **Yes**

Auto-restart MySQL: [?] **No**

MySQL Root Password (All fields are required)

Password:

Confirm Password: [?]

MySQL Server Options (All fields are required)

Connections (max): [?]

Connection Errors (max): [?]

Connect Timeout: [?]
In Seconds

Wait Timeout: [?]
In Seconds

Key Buffer Size: [?]
In Bytes

Sort Buffer Size: [?]
In Bytes

Read Buffer Size: [?]
In Bytes

Maximum Allowed Packet Size: [?]
In Bytes

Thread Cache Size: [?]

Table Cache: [?]

Query Cache Limit: [?]
In Bytes

Query Cache Size: [?]
In Bytes

Current MySQL Processes [?]

ID	Username	Command	Time
NO ACTIVE MYSQL PROCESSES / QUERIES			

Figure 2.1: The MySQL Overview Page

Basically, the setting is a security setting that is designed to combat brute-force password guessing attacks. Most hosts don't enforce it because none of the users are permitted remote login, or TCP port 3306 is blocked in the firewall. If you have clients connecting direct to your MySQL server remotely though, you may want to lower this to a reasonable amount to prevent brute-force login attempts. The default is 10.

2.1.3 Connect Timeout

This specifies how long it takes for the server to wait on a *connect packet* before the server deems that the connection has timed out. This value is normally not that critical but it may be helpful to modify it if clients frequently encounter errors of the form: `Lost connection to MySQL server at 'XXX', system error: errno`. The default MySQL value is 10.

What does it mean to wait for a *connect packet*? A connection to a MySQL server requires 2 things - a TCP connection, and then via MySQL's protocol, a request to connect. Connect timeout will become a factor only when a remote client has initiated a TCP connection to port 3306 - but has not sent a request to connect to the MySQL server proper. In layman's terms: the MySQL server is aware that there is some computer connected to it, but it will not consider it a client to its service until the remote computer says "I know you are a MySQL server, I want to connect to your service!". If the remote computer fails to provide this within the connect timeout window, MySQL will drop the connection.

2.1.4 Wait Timeout

This specifies how long a MySQL connection can remain open with nothing happening on it before the server kills the connection. If you for some reason need to remain connected to your MySQL server for long periods of time, you may want to increase this. On the other hand, consider that a connection open is eating up one of your `max_connections` slots. You may want to lower the timeout to prevent a bunch of users from sitting connected for a lengthy period of time. The default is 28800 seconds.

2.1.5 Key Buffer Size

When defining MySQL tables, you occasionally may define one of the columns as a Primary Key, Unique, or Index column. I show an example of this in Figure 2.2. You may do it for the ability to put restrictions on what rows you


```
CREATE TABLE users (
    userid integer ,
    Last_Name varchar(30),
    First_Name varchar(30),
    PRIMARY KEY (userid)
);
```

Figure 2.2: An example of creating a table with a primary key

can insert into your table. I.e if I have a column for “userid” I don’t want 2 different users with the same userid, so I would set the column to be a primary key or unique so that the database system doesn’t allow me to accidentally insert a record with an identical ID.

The column key designation also has the additional effect of making data look-ups faster. For example, if I have a column *userid* in a table *users*, and I have 1 million users in my database, a query like `SELECT * FROM users WHERE userid=10` will be trivial for MySQL to perform because each row in the table will be “optimized” to be looked up by the *userid*³. In order to speed things up even more, commonly used meta data⁴ that is used by the server to find rows quickly is cached in memory. This is known as the `KEY BUFFER` or `KEY CACHE` and you can control it’s size. By default, most users will want to stay with what ever the key buffer size is on operating install. But if you are more adventurous or you have a lot of memory that you can devote to the MySQL server, changing the value may be in your interest. To give more detailed information on modifying this value, we will quote the MySQL documentation again:

The maximum permissible setting for `key_buffer_size` is 4GB on 32-bit platforms. As of MySQL 5.0.52, values larger than 4GB are permitted for 64-bit platforms (except 64-bit Windows, for which large values are truncated to 4GB with a warning). The effective maximum size might be less, depending on your available physical RAM and per-process RAM limits imposed by your operating system or hardware platform. The value of this variable indicates the amount of memory requested. Internally, the server allocates as much memory as possible up to this amount, but the actual allocation might be less.

You can increase the value to get better index handling for all reads and multiple writes; on a system whose primary function is to run MySQL using the MyISAM storage engine, 25% of the machine’s total memory is an acceptable value for this variable. However, you should be aware that, if you make the value too large (for example, more than 50% of the machine’s total memory), your system might start to page and become extremely slow. This is because MySQL relies on the operating system to perform file system caching for data reads, so you must leave some room for the file system cache. You should also consider the memory requirements of any other storage engines that you may be using in addition to MyISAM.⁵

It should be noted here that your InterWorx server is not *only* running MySQL and dedicating 25% of your system’s memory to the MySQL key cache is not recommended. 25% should probably be the maximum memory footprint that the entire MySQL service takes up.

2.1.6 Sort Buffer Size

The sort buffer is a per-session (read: per connection) buffer that is created on demand when sorting or grouping needs to happen in order to process a query. In SQL this translates to when `ORDER BY` or `GROUP BY` are used to try and filter/sort data. By default this value is set to 2MB and it is generally recommended to keep this value at default or within 256KB-4MB since this is a *per session* cache. 100 concurrent sessions doing heavy sorting would eat 200MB of memory solely for sorting (not including various other caches). In general, MySQL recommends that if you have slow queries with `ORDER BY` or `GROUP BY` in them, you should investigate better query optimization or improved indexing instead of messing with the sort buffer size first. The maximum size for the sort buffer is 4GB, but again - you don’t want huge sort buffers.

³This is because on the back-end, the primary key is fed into a hashing algorithm to find the correct row in the file that backs your database.

⁴index blocks which more than likely contain hash tables for rapid look-ups of specific rows based on keys

⁵http://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html#sysvar_key_buffer_size

2.1.7 Read Buffer Size

Any MySQL query that needs to do a sequential (read: in order) scan of a given table needs somewhere to store in memory what it has read thus far as it traverses down the table. This is the read buffer and its size is controlled by the `read_buffer_size` option. By default MySQL sets the value to 131072 bytes, or 128Kbytes. The thing to note about the read buffer size is that the byte size should be a multiple of 4KB or 4096 bytes, the default page size of most operating systems on x86. If the value entered is not a multiple of 4KB, it will be rounded down to the nearest multiple when the server daemon is started. You may want to increase this value if you have extra memory and some of the tables that are getting scanned are quite big.

2.1.8 Maximum Allowed Packet Size

While packets or frames have different meanings in the context of Ethernet and TCP/IP protocols, in MySQL land the communication packet refers to the set of data that a client issues to a server when making an SQL query, or the set of data corresponding to a single table row being sent back to the client. For the most part, the maximum allowed packet size is not something that is typically going to be modified by most hosts since the default of 1MB is sufficient in most cases. On the other hand, certain applications may make use of the BLOB or TEXT column type, which can accept very large data inputs. If you have clients storing large sets of data in their database columns and they are getting errors such as “Packets larger than `max_allowed_packet` are not allowed” when they attempt to insert or select large table rows, you will want to increase this value to accommodate that data set.

2.1.9 Thread Cache Size

In order to concurrently serve the SQL queries of multiple applications and MySQL clients at once, the MySQL server uses threads. For the non-programmer, a thread is like a mini processes that runs inside the process of the MySQL daemon. It basically operates independent of the execution of other threads, including the one that was started with the program. This allows for the MySQL daemon to concurrently handle multiple connections and queries simultaneously. When a new connection comes in, MySQL (and possibly the operating system) has to expend some CPU time and memory to initialize a new thread to handle the session. The the thread cache allows MySQL to save some threads instead of just letting them die at the end of handling a connection so that the next connection can be serviced quicker.

This variable is typically only really worth modifying if you know you have hundreds of new connections per second and you are seeing a huge performance hit. The default is 0, connection threads are not cached.

2.1.10 Table Cache

This is the number of tables that MySQL can have open concurrently. It is important to note that MySQL will “open” a table once per connection - that means that even if you have a single table in your database, if 100 connections need that table, MySQL will attempt to open that table up to `table_cache` times.

Table cache is lightly related to the `Connections(Max)` field as described in section 2.1.2. Basically if you are expecting a certain number of connections, let’s say for example 100, you want table cache to be set to $100 * N$ where N is the maximum number of tables you are JOIN-ing together in a given query. Understandably, this value is hard to guess and as such, it is safe to say on a default InterWorx install, you can leave this value unchanged until either the concurrent number of connections on your MySQL server increases, or you notice that queries going through are JOIN-ing a significant number of tables.

Another thing to bear in mind with the Table Cache is that each open table requires an additional Linux file descriptor. Occasionally operating system security, VPS security (if you are inside a VPS), or even MySQL’s `open-files-limit` can cap the number of file descriptors that are allowed to be open by the MySQL daemon process. If MySQL hits this limit, it will “panic” and disallow any new connections. The MySQL documentation says the following:

“Make sure that your operating system can handle the number of open file descriptors implied by the `table_cache` setting. If `table_cache` is set too high, MySQL may run out of file descriptors and refuse connections, fail to perform queries, and be very unreliable. You also have to take into account that the MyISAM storage engine needs two file descriptors for each unique open table. A MyISAM table is opened for each concurrent access. This means the table needs to be opened twice if two threads access the same table or if a thread accesses the table twice in the same query (for example, by joining the table

to itself). Each concurrent open requires an entry in the table cache. The first open of any MyISAM table takes two file descriptors: one for the data file and one for the index file. Each additional use of the table takes only one file descriptor for the data file. The index file descriptor is shared among all threads.”⁶

The cache of open tables is kept at a level of TABLE CACHE entries. The default value is 64. Note that MySQL may temporarily open more tables than this to execute queries. MySQL closes an unused table and removes it from the table cache under the following circumstances:

- When the cache is full and a thread tries to open a table that is not in the cache.
- When the cache contains more than table_cache entries and a table in the cache is no longer being used by any threads.
- When a table flushing operation occurs. This happens when someone issues a FLUSH TABLES statement or executes a mysqladmin flush-tables or mysqladmin refresh command.

When the table cache fills up, the server uses the following procedure to locate a cache entry to use:

- Tables that are not currently in use are released, beginning with the table least recently used.
- If a new table needs to be opened, but the cache is full and no tables can be released, the cache is temporarily extended as necessary. When the cache is in a temporarily extended state and a table goes from a used to unused state, the table is closed and released from the cache.

2.1.11 Query Cache Limit

By default, MySQL will try to cache the results of a SELECT statement so that if the query is issued often and the data backing the query doesn't change, the expensive computation needed to return the query doesn't have to be performed a second time. Most applications change their databases less often than how often data is requested from the database. As such, MySQL is able to return the result of query much faster if it has the result already sitting in memory.

The QUERY LIMIT CACHE controls what the maximum size of a result is that is allowed to go into a cache. By default, this is set to 1MB. You can increase this to improve performance on queries which return large data sets, but you do so at the risk of significantly increasing the memory footprint of the MySQL server.

2.1.12 Query Cache Size

The QUERY CACHE SIZE variable controls the size to allocate to query caching on the MySQL server. This value must be a multiple of 1024 bytes, and must be at minimum of 40KB to allocate the data structures required for caching. According to MySQL, the default is 0 (as in, no caching will occur). 64-256MB is not unreasonable, though, depending on your resources available.

2.2 Additional features of Overview Page

In addition to manipulating settings found in the /etc/my.cnf MySQL configuration file, this page also allowed you to set the MySQL root password, and view active processes.

2.2.1 The MySQL Server Root Password

A common question InterWorx gets is - what is my MySQL server's root password? During installation, InterWorx connects to the MySQL database using the MySQL root user since the root user should not have a password after Operating System install. InterWorx creates a special iworx user that has all root privileges which is what InterWorx will use from that point on to interact with the MySQL server. InterWorx then sets a random, scrambled, gibberish password for the MySQL root user so that it is not accessible without a password anymore.

Therefore, by default, no one knows your root MySQL password. Since InterWorx's iworx user has root privileges, InterWorx is able to set your root password if you wish to access the database by that user. This can be done through MySQL root password box, visible in figure 2.1.

⁶<http://dev.mysql.com/doc/refman/5.0/en/table-cache.html>

Value	Meaning
Binlog Dump	This is a thread on a master server for sending binary log contents to a slave server.
Change user	The thread is executing a change-user operation.
Close stmt	The thread is closing a prepared statement.
Connect	A replication slave is connected to its master.
Connect Out	A replication slave is connecting to its master.
Create DB	The thread is executing a create-database operation.
Daemon	This thread is internal to the server, not a thread that services a client connection.
Debug	The thread is generating debugging information.
Delayed insert	The thread is a delayed-insert handler.
Drop DB	The thread is executing a drop-database operation.
Error	Bad things.
Execute	The thread is executing a prepared statement.
Fetch	The thread is fetching the results from executing a prepared statement.
Field List	The thread is retrieving information for table columns.
Init DB	The thread is selecting a default database.
Kill	The thread is killing another thread.
Long Data	The thread is retrieving long data in the result of executing a prepared statement.
Ping	The thread is handling a server-ping request.
Prepare	The thread is preparing a prepared statement.
Processlist	The thread is producing information about server threads.
Query	The thread is executing a statement.
Quit	The thread is terminating.
Refresh	The thread is flushing table, logs, or caches, or resetting status variable or replication server information.
Register Slave	The thread is registering a slave server.
Reset stmt	The thread is resetting a prepared statement.
Set option	The thread is setting or resetting a client statement-execution option.
Shutdown	The thread is shutting down the server.
Sleep	The thread is waiting for the client to send a new statement to it.
Statistics	The thread is producing server-status information.
Table Dump	The thread is sending table contents to a slave server.

Table 2.1: Values for the “Command” field and their meanings

2.2.2 Current MySQL Processes

This table will show you the current snapshot of MySQL processes which is done by running `SHOW PROCESSLIST` as the `iworx` user (which has root privileges as described in section 2.2.1).

ID is the the “connection identifier”, or how MySQL internally references the connection that issued the query.

User is the MySQL user that is currently connected. This can be particularly if you have many SiteWorx accounts and you are trying to track down particularly resource abusive sites. Users with high process run time and many processes in this list will probably be issuing extremely inefficient queries. Alternatively, their data sets may have grown large enough that some tweaking needs to be performed in order to improve server performance.

Command is the type of command the thread is currently executing. Keep in mind that this is not analogous to what query was executed, but what “internal” procedure the thread is performing. They can take on the values seen in table 2.1. Essentially this is the “state” the thread is in. Most of the time, threads are busy in the Query state. With the popularity of using prepared statements, though, you may see other states occasionally on a busy server.

Time is the amount of time in seconds a thread has been in its current state.

MySQL Server Management

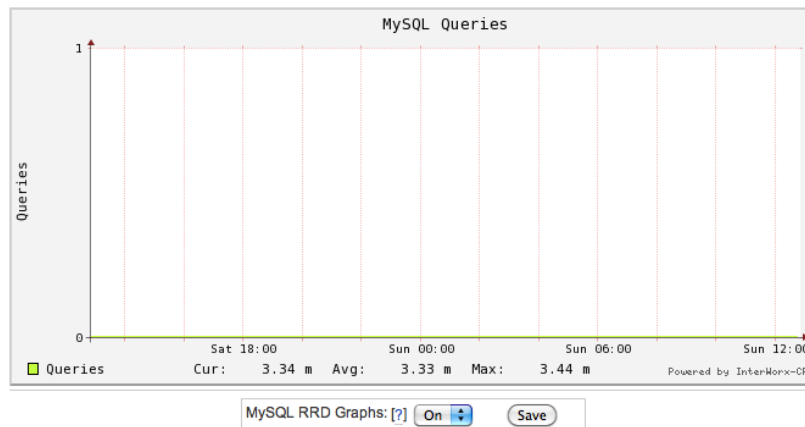


Figure 2.3: The MySQL RRD Graph

2.2.3 RRD Graph

As a web host, you often want to track basic statistics of your server in order to determine if work load or resource usage has changed over time. Also on the MySQL overview page is a graph of the average number of MySQL queries over time, with the X-axis representing time and the Y-axis representing the number of queries. A picture of a (blank) graph can be seen in figure 2.3. This can be useful if you notice your MySQL server has been using more resources lately - the graph can demonstrate whether that is due to an increase query workload or not.

Chapter 3

Remote Servers

The remote servers feature, as stated earlier, allows you as a web host to remove the responsibility of running a MySQL database server from the InterWorx server and move it to it's own segregated server. Furthermore, if one MySQL server becomes overwhelmed, you can additional MySQL servers for new accounts to use. Remote servers can be managed in NodeWorx under System Services > MySQL Server > Remote Servers

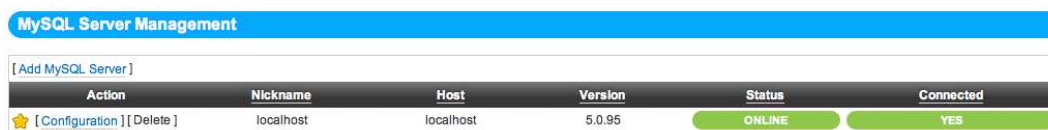
3.1 The Remote Servers Screen

As seen in Figure 3.1, the remote servers screen lists all the servers, including the one running locally on the InterWorx box (localhost). The main purpose of this screen is to give you a quick look at all the servers available for you and your resellers to assign SiteWorx accounts to, as well as their status (Online/Offline/Connected/Disconnected) and version. This screen also allows you to set what the “default” MySQL server, and remove remote servers if you no longer want them available to your InterWorx server. By default, when you visit this screen all you’ll see is the **localhost** server. *Keep in mind that changing MySQL servers of a SiteWorx account after it’s created is not a trivial process as data may need to be migrated, and version discrepancies may create issues.* See chapter 7 for details on how this is done.

3.2 Adding a remote server

As discussed in section 1.1 and section 1.2.2, the requirements for a remote server are:

- The MySQL server version must be between 4 and 5.5.
- In the /etc/my.cnf, the `old-passwords` setting must be turned on (set to true). This is because the MySQL client that InterWorx uses to interact with the system MySQL server is a 4.0 client.
- You must know the root MySQL password for InterWorx to interact with the server.
- The InterWorx server must be able to communicate with the MySQL server on TCP port 3306.
- The remote MySQL server must be setup to receive incoming connections, particularly for the root MySQL user.



Action	Nickname	Host	Version	Status	Connected
[Add MySQL Server] [Configuration] [Delete]	localhost	localhost	5.0.95	ONLINE	YES

Figure 3.1: The Remote Servers screen

Figure 3.2: Adding a new remote MySQL server

Action	Nickname	Host	Version	Sta
★ [Configuration] [Delete]	localhost	10.1.10.166	5.1.61	ONL
★ [Configuration] [Delete]	db2	10.1.10.169	5.1.61	ONL

Figure 3.3: Setting a default MySQL server

As you can see in figure 3.2, adding a new server is fairly straight forward. You need to pick a “nickname” for the MySQL server which will be used by InterWorx to refer to that server easily instead of using host name or IP. This also makes it easier when adding new accounts through our API or CLI because you specify the MySQL server to use via nickname. It also is easier to remember what server is for what when you are adding new MySQL servers by IP address.

3.3 Setting a default MySQL server

As you can see in figure 3.3, setting a default server is as simple as clicking a “favorite” star next to the preferred server. By setting a default MySQL server, you are setting the server that will be used automatically when creating a SiteWorx account without an explicitly set remote server. Most billing systems interact with InterWorx on a very basic level, and many don’t go out of their way to specify which MySQL server to use since that is not the concern of most billing systems. By setting a default MySQL server, you are ensuring that unless otherwise specified, a SiteWorx account will be mapped to the correct server.

Another benefit of the default server setting is when creating a SiteWorx account manually through the interface, the default server will be selected by default in the drop down menu.

3.4 Deleting a remote server

Deleting a server from InterWorx is quite easy, you simply click delete and the system will prompt you to make sure you are sure you want to remove the server. By default, InterWorx will not permit you to remove MySQL servers that are currently in use by SiteWorx accounts. Once deleted, you will not be able to map SiteWorx accounts to that database and if it was a default server, the default will return to localhost. You can never delete the localhost server, only turn it off.

3.5 Managing/Configuring a remote server

Unfortunately InterWorx isn't magical and can't change `/etc/my.cnf` settings on the other box without opening an SSH session. Therefore, when clicking [Configuration] on the Remote Servers screen, you will see a simplified version of the overview page for the localhost server which details the status of the server (up or down), the ability to change root password, and also active threads/processes. Essentially, you are limited to what can be accessed via the MySQL client. In order to manipulate settings, you will have to do so outside of InterWorx directly on the server itself. The [MySQL documentation](#) is a fantastic resource that can guide you on configuring your remote server.

Chapter 4

phpMyAdmin

This will be a rather brief chapter for 2 reasons:

1. phpMyAdmin is rather user friendly and intuitive to use.
2. phpMyAdmin has it's own documentation if you actually are stuck.

4.1 Basics

phpMyAdmin is a nice web-based front end to a MySQL server and its databases. By default, it only shows you the databases that the MySQL user you are logged in as is permitted to see. A screen shot is shown in figure 4.1.

4.1.1 phpMyAdmin inside NodeWorx

When using phpMyAdmin in NodeWorx, you are logged in as the `iworx` MySQL user, which has root privileges. As a result, you will be able to see all databases and tables on your MySQL server, as well as modify them freely.

In the event that you have remote servers or you have debugging mode enabled¹, when clicking the phpMyAdmin link, you will be taken to a landing page which will display a drop down box of all the database servers that InterWorx can interface with. The reason debugging mode causes this to occur is because debugging mode will allow you to see the control panel's database server (the separate, segregated server that the control panel uses to store NodeWorx and SiteWorx meta data). This is useful if you are a more technical user and have enough experience to look at the InterWorx database. *We strongly discourage modifying anything directly in the InterWorx database without guidance from InterWorx support. This could lead to unexpected results and or serious issues for your server.*

4.1.2 phpMyAdmin in SiteWorx

In SiteWorx, phpMyAdmin operates a bit differently. Since there can be multiple MySQL users in a SiteWorx account, we can't just log the user in as a given user and expect them to see all their databases. Instead, InterWorx creates a temporary MySQL user on demand which has access to all the databases of a given SiteWorx account.

phpMyAdmin can be an invaluable tool for debugging issues with web applications, fixing data integrity issues, importing large data sets, exporting large data sets, creating new databases, and designing the database structure.

4.2 Common Session Issue

The most common issue we see is that NodeWorx admins (read: users who are able to log in and out of different SiteWorx accounts) often will go into phpMyAdmin in multiple accounts, or go into phpMyAdmin in NodeWorx, then log into a SiteWorx account and go to phpMyAdmin in SiteWorx *without explicitly clicking the logout button in phpMyAdmin*. Since phpMyAdmin is opened in a tab, often users will simply close their tab instead of explicitly

¹Debugging mode can be activated in NodeWorx under Server > Settings

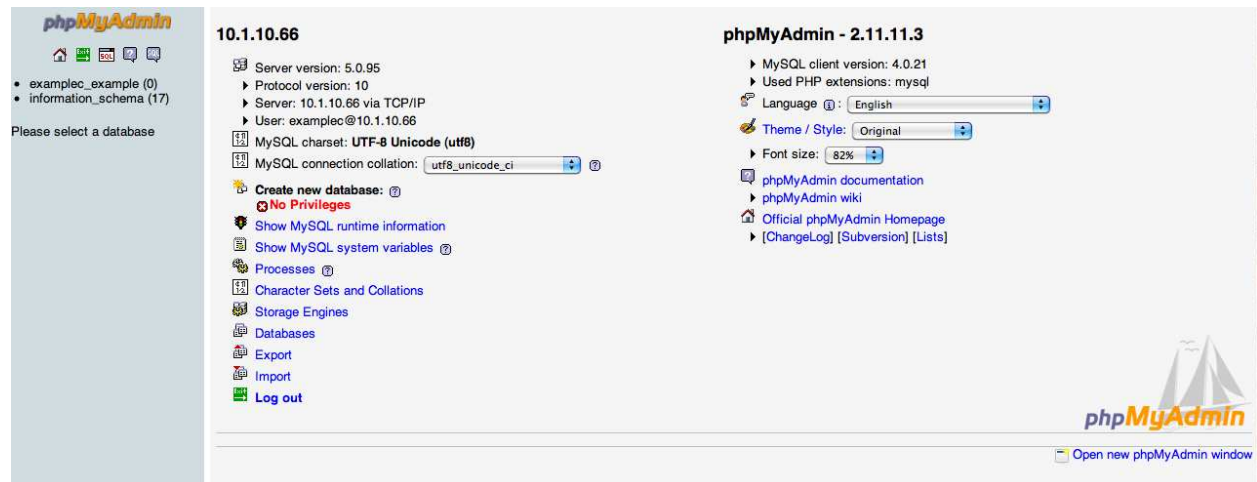


Figure 4.1: phpMyAdmin

clicking logout. While this does not matter for a SiteWorx user with multiple accounts, occasionally NodeWorx administrators who move between SiteWorx account's phpMyAdmin or move between NodeWorx phpMyAdmin and SiteWorx phpMyAdmin without explicitly clicking the green logout button will see that the session from the last-used phpMyAdmin session will appear instead of the one they were expecting.

InterWorx tries to mitigate reloading and re-creation of the temporary SiteWorx MySQL accounts unnecessarily, and thus it defaults to using the last session, if it's still valid. If you have not closed your browser and you switch between phpMyAdmin's, it will appear that a SiteWorx account might be seeing someone else's data when in reality it is only your web browser (as the NodeWorx server admin) that has the capability (temporarily). This can be fixed by clicking the green logout button in phpMyAdmin, (as that will explicitly flush the phpMyAdmin session), and clicking the phpMyAdmin button again. Alternatively, you can just close your web browser and re-open it.

Part II

Advanced MySQL Administration and Management

Chapter 5

Tracking down load issues

Overtime as you add more accounts and more users on your system, occasionally you'll get the few bad apples with poorly-coded web applications that hammer your MySQL server with LEFT JOIN's with SORT BY's and GROUP BY's on one million row tables that completely obliterates your server. If you have a large number of SiteWorx accounts, tracking down which one is responsible for MySQL sucking up your server's resources may prove difficult. The tool that InterWorx support will typically use is called **mytop**. Mytop is a Perl application that continuously queries the MySQL server with SHOW PROCESSLIST and produces produces output anolgous to the linux system load monitoring tool, top. This allows you to see a somewhat-in-real-time picture of what's going on with your server.

5.1 Installing mytop

1. You can download mytop from [the MyTop homepage](http://jeremy.zawodny.com/mysql/mytop/mytop-1.6.tar.gz). You can just download it to your /root/ folder. You can run:
`wget HTTP://jeremy.zawodny.com/mysql/mytop/mytop-1.6.tar.gz`
2. Extract the tarball with:
`tar xvzf mytop-1.6.tar.gz.`
3. Try running it by cd'ing to the directory that was just created and running:
`./mytop.`
4. If you get a warning about Term::ReadKey not being installed, we will need to try and install it via CPAN (which is like a package manager for perl modules).
5. Check if you have CPAN by just running `cpan` on your terminal. If not, try installing it with:
`yum install perl-CPAN.`
6. Once you have run `cpan`, CPAN might ask you if you want it to configure itself. Simply hit enter to have CPAN automatically configure itself.
7. Once at the prompt, enter `install Term::ReadKey` in order to have cpan download and install the package. Then enter `quit` to exit.
8. At this point you should be good to go. You may be missing other dependencies, most of which should be obtainable via CPAN if needed.
9. If you get an error when running mytop: `'Error in option spec: "long!!"'`, then you should open the `./mytop` file in your preferred text editor, go to the line that looks like:
`"long!!" => \${config{long_nums}},`
and comment it out with a `#` character in front of that text. On our version (1.6), the line was number 159.

```

MySQL on mysql4.inf.dcn (4.0.15-Yahoo-SMP) up 2+20:12:18 [10:07:27]
Queries: 1.9M qps: 8 Slow: 0.0 Se/In/Up/De(%): 00/95/00/00
          qps now: 1 Slow qps: 0.0 Threads: 53 ( 1/ 0) 00/00/00/00
Key Efficiency: 100.0% Bps in/out: 1.5k/740.2 Now in/out: 321.4/ 2.5k

  Id      User      Host/IP      DB      Time      Cmd Query or State
  --      -
  62      jzawodn    autopsy     mysql    0      Query show full process
  8       root      localhost    test     1      Sleep
  63      jzawodn    mysql2      mysql    2      Sleep
  64      jzawodn    mysql2      mysql    2      Sleep
  65      jzawodn    mysql2      mysql    2      Sleep
  66      jzawodn    mysql2      mysql    2      Sleep
  67      jzawodn    mysql2      mysql    2      Sleep
  68      jzawodn    mysql2      mysql    2      Sleep
  69      jzawodn    mysql2      mysql    2      Sleep
  70      jzawodn    mysql2      mysql    2      Sleep
  71      jzawodn    mysql2      mysql    2      Sleep
  72      jzawodn    mysql2      mysql    2      Sleep
  73      jzawodn    mysql2      mysql    2      Sleep
  74      jzawodn    mysql2      mysql    2      Sleep
  75      jzawodn    mysql2      mysql    2      Sleep

```

Figure 5.1: An example mytop display from <http://jeremy.zawodny.com/mysql/mytop/>

5.2 Using mytop and logging into the MySQL database as root

In order to use mytop, you need to provide server and login information. Typically most users don't know their iworx MySQL password to the localhost database. We recommend instead manually setting the root password in NodeWorx under System Services > MySQL Server > Overview for your localhost MySQL server. If you need to set the root password for a remote server, this can also be accomplished by visiting System Services > MySQL Server > Remote Servers and clicking [Configuration] next to the remote server you wish to set the password for.

5.2.1 Connecting to your database server with mytop

In order to connect with mytop now, you can run the following:

```
./mytop -S /var/lib/mysql/mysql.sock -u root -p [PASSWORD HERE] -d mysql
```

The `-d mysql` is added because a database needs to be selected. As all servers have a `mysql` database (it stores meta data like users and permissions), that is a safe database to choose. As root user, though, you will see all activity across all databases and users.

5.3 mytop Display (From Official Documentation)

This documentation can be found at <http://jeremy.zawodny.com/mysql/mytop/mytop.html>. The mytop display screen is really broken into two parts and is shown in figure 5.1. The top 4 lines (header) contain summary information about your MySQL server.

- The first line identified the host name of the server (localhost) and the version of MySQL it is running. The right hand side shows the uptime of the MySQL server process in days+hours:minutes:seconds format (much like FreeBSD's top) as well as the current time.
- The second line displays the total number of queries the server has processed, the average number of queries per second, the real-time number of queries per second, and the number of slow queries.
- The third line deals with threads. Versions of MySQL before 3.23.x didn't give out this information, so you'll see all zeros.
- And the fourth line displays key buffer efficiency (how often keys are read from the buffer rather than disk) and the number of bytes that MySQL has sent and received.

You can toggle the header by hitting h when running mytop. The second part of the display lists as many threads as can fit on screen. By default they are sorted according to their idle time (least idle first). As you can see, the thread id, username, host from which the user is connecting, database to which the user is connected, number of seconds of idle time, the command the thread is executing, and the query info are all displayed. Often times the query info is what you are really interested in, so it is good to run mytop in an xterm that is wider than the normal 80 columns if possible. The thread display color-codes the threads if you have installed color support. The current color scheme only works well in a window with a dark (like black) background. The colors are selected according to the Command column of the display:

- Query - Yellow
- Sleep - White
- Connect - Green

Those are purely arbitrary and will be customizable in a future release. If they annoy you just start mytop with the -nocolor flag or adjust your config file appropriately.

Instead of always using bulky command-line parameters, you can also use a config file in your home directory (~/.mytop). If present, mytop will read it automatically. It is read before any of your command-line arguments are processed, so your command-line arguments will override directives in the config file.

Here is a sample config file ~/.mytop which implements the defaults described above:

```
user=root
pass=
host=localhost
db=test
delay=5
port=3306
socket=
batchmode=0
header=1
color=1
idle=1
```

Using a config file will help to ensure that your database password isn't visible to users on the command-line. Just make sure that the permissions on ~/.mytop are such that others cannot read it (unless you want them to, of course). You may have white space on either side of the = in lines of the config file.

Shortcut Keys The following keys perform various actions while mytop is running. This list has been abridged to those keys that will probably be helpful during trouble-shooting load issues.

- **?** Display help.
- **d** Show only threads connected to a particular database.
- **f** Given a thread id, display the entire query that thread was (and still may be) running.

- **F** Disable all filtering (host, user, and db).
- **i** Toggle the display of idle (sleeping) threads. If sleeping threads are filtered, the default sorting order is reversed so that the longest running queries appear at the top of the list.
- **k** Kill a thread.
- **m** Toggle modes. Currently this switches from ‘top’ mode to ‘qps’ (Queries Per Second Mode). In this mode, mytop will write out one integer per second. The number written reflects the number of queries executed by the server in the previous one second interval. More modes may be added in the future.
- **o** Reverse the default sort order.
- **p** Pause display.
- **q** Quit mytop
- **s** Change the sleep time (number of seconds between display refreshes).
- **u** Show only threads owned by a given user.

5.4 General Paradigm for Tracking Load Issues

In general, you want to look for either:

1. Queries which have been running for an excessively long time (high Time value)
2. Multiple threads mostly owned by the same user.

The combination of these two can often help sort out the nature of what’s going on with load. If a user has excessive queries, they might be getting a lot of traffic and it might be time to segregate that user on their own MySQL server, or even move them to a less busy server with more resources. If you are seeing long-running queries, you may want to try and work with the host to improve their application’s code base, or alternatively ask them to pay more for their resource use. Again, how you handle situation with single SiteWorx accounts depends on your policy and service agreement with your client. This guide’s purpose is to help you root the cause of high MySQL usage.

Chapter 6

Data backup, repair, recovery

Databases are often the most valuable part of an application on the internet. Unfortunately, databases and the data they maintain are subject to failure. The hardware backing the filesystem your database run on could die, power could be cut and the hard disk cache might not be able to write cached data back to the magnetic platters, the filesystem might be unable to figure out what is missing and make entire files “disappear”, the raid controller might die, and you get the idea. Even though newer MySQL datastores are more reliable, things can still go awry. This chapter deals primarily with 2 things:

1. How you can have InterWorx backup MySQL data for you
2. How to try and recover when your database becomes corrupted

6.1 Backup inside InterWorx

The interface that InterWorx provides allows you to generate backups of SiteWorx accounts, which by default contain a dump of all the databases tied to this account. The SiteWorx interface also allows you to make partial backups that contain only MySQL database dumps. These partial backups can be used to take snapshots of valid database states. It is important to keep in mind that if your database is corrupted, the process which dumps your database will be unable to make a database dump and the backup process will fail. This is both to your advantage (if the database is corrupted you aren't taking snapshots of bad data) and disadvantage (if you aren't paying attention to the alert emails, you might miss that automated backups are failing).

6.1.1 From NodeWorx

Inside NodeWorx under SiteWorx > Backup / Restore, you have the ability to do bulk backups of multiple SiteWorx accounts. This is shown in figure 6.1. This is good if your clients aren't very prone to backing up their content

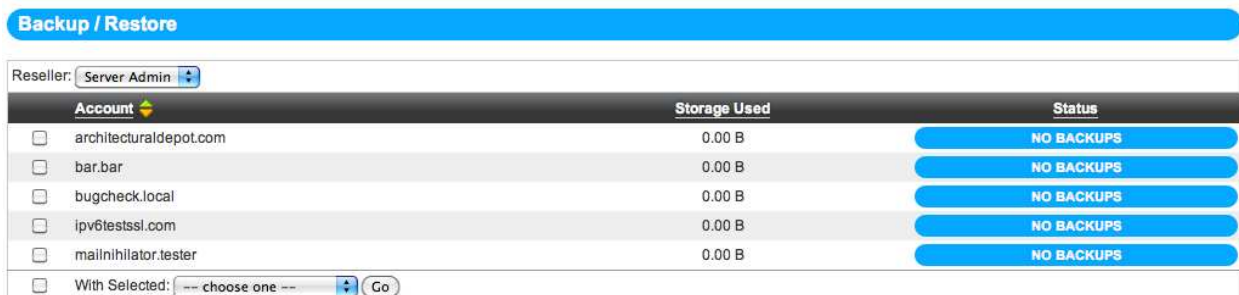


Figure 6.1: NodeWorx Backup/Restore Screen

Managing Domain: [example.com](#)

Additional input is required

Additional input is required (All fields are required)

Backup Type:	Full Backup
Backup Where:	Default Location

Continue Cancel

Figure 6.2: The Backup Now starting prompt

regularly. Select the SiteWorx accounts you wish to backup, and on the “With Selected:” drop down, select “Full-Backup”. “Structure-only” will not backup the databases and thus is not helpful when trying to preserve MySQL data.

6.1.2 From SiteWorx

From inside SiteWorx accounts, you are given much more control of the backups and are able to create database-only backups. In addition you can schedule backups to occur regularly which allows for a hands-free approach to backing up.

6.1.2.1 Backup Now

Inside a SiteWorx account, under Backups ▸ Backup Now, you can create a backup on demand for your database, with the additional ability to have it sent to a remote server when the backup process completes. (Backups are tar’d and gzip’d which might take some time depending on the size of the database). As you can see in figure 6.2, you are asked for 2 things when you Backup Now.

Backup Type is how complete of a backup do you want for the SiteWorx account. In this case, you want to choose is “Partial Backup”. Don’t worry, you will be asked next what you want to backup.

Backup Where is where you want the completed file to be stored. The default location is `/home/[SiteWorx linux user]/[active domain]/iworx-backup`. Backups placed in this directory are recognized by InterWorx when attempting a restore from NodeWorx. The alternatives to default location are:

Local File: You don’t want InterWorx to name and place the file for you, you want to dictate filename and where the backup ends up.

FTP: The file will be FTP’d to another server on completion. You need to know FTP login credentials and the path structure of the remote server.

SFTP/SCP: The SCP command will be used to send the file to another server. You need to know a linux user login on the remote server to use this.

The next step of the process is shown in figure 6.3. As you selected partial backup in the previous step, you are now prompted for what specifically you want to backup from the site. Since we are interested in preserving databases, “Backup your databases” should be checked. Depending on what you selected for “Backup Where” with the exception of Default Location, you will be prompted for paths and login credentials for sending the completed backup file to its storage location. Clicking the Backup button will begin the backup process and you will be notified via email (based on what you put in “Email Status To”) when the backup completes successfully or if there’s an issue of some sort.

6.1.2.2 Schedule

Setting up a scheduled backup is similar to Backup Now, as discussed in the previous section, with the additional responsibilities of electing backup frequency, and how many backups to store at once (old ones are removed). Scheduling backups is done in SiteWorx under Backups ▸ Schedule. A screen shot of the dialog that appears when you click [Create Scheduled Backup] is shown in figure 6.4. Once you select your desired backup frequency, a dialog similar to what

Create Backup

Create Backup (All fields are required)

E-mail status to: ✓

Backup Type

Backup Type: Partial Backup

What To Backup:

- Backup your website
- Backup your e-mail
- Backup your databases

Backup Where

Backup Where: FTP

Path: ✓

Username: ✓

Password: ✓

Confirm Password: [?] ✓

Hostname: ✓

Port: ✓

Passive Mode: ▾

Figure 6.3: Backup Now with Partial Backup as Backup Type and FTP for Backup Where

Additional input is required

Additional input is required (All fields are required)

Frequency: ▾

Backup Type: ▾

Backup Where: ▾

Figure 6.4: Scheduling a Backup Screen

Figure 6.5: Setting up a scheduled backup

you saw in Backup Now appears. This is shown in figure 6.5. As you can see, you are able to set the maximum number of backups to store at a given time. As a NodeWorx admin, you are able to control the max limit in NodeWorx, and this is covered in depth in the Backup guide. Setting this value to something reasonable will ensure that you don't run out of disk space on the storage medium that will hold the backup.

6.1.2.3 Using phpMyAdmin to backup

The nice thing about phpMyAdmin is that it has some great capabilities for generating backups on the fly through an interface instead of command-line. If you just want a raw MySQL dump of your database, you may elect to take this option for convenience. Under Hosting Features > MySQL > PhpMyAdmin, you can get access to the phpMyAdmin for the current SiteWorx account. Figure 6.6 shows where you can find the "Export" link in phpMyAdmin. From the export page you can determine the exact characteristics of the SQL dump of your database(s). The export page is shown in figure 6.7. Let's walk through all the settings on this page, in case you aren't feeling comfortable with all that is there.

Export This box contains 2 choices, the databases you want to export, and what format the export should be in. For database selection, you probably want to backup everything with the exception of the information_schema table. The information_schema is a meta-data table that MySQL maintains to make statistics and information about databases available to the user. It is not actually stored on disk anywhere - any SELECTS on this table have results generated on-demand by the MySQL server. Thus, backing it up is somewhat silly.

For export type, you will probably want to go with SQL for backup purposes. These are the raw SQL commands which would re-generate your database and data on the fly. Other formats might be really convenient for examining data by human eye or exporting your data to excel sheets for statistics analysis.

Options Let's go down the list of options one-by-one:

Add custom comment into header allows you to add a special comment to your SQL output. It is possible to make comments - i.e. non-executable SQL statements - inside SQL files and the export function will gratuitously add comments to make the SQL dump easier to read for humans. If you wish, you can add a special comment at the top with some information important to you.

Enclose export in a transaction only matters if you are using InnoDB as your storage engine for your database tables. By default, tables are created using MyISAM as the storage engine which does not support the transaction model. Transactions allow you to guarantee that the data you get is in a "balanced" state. For example, if when

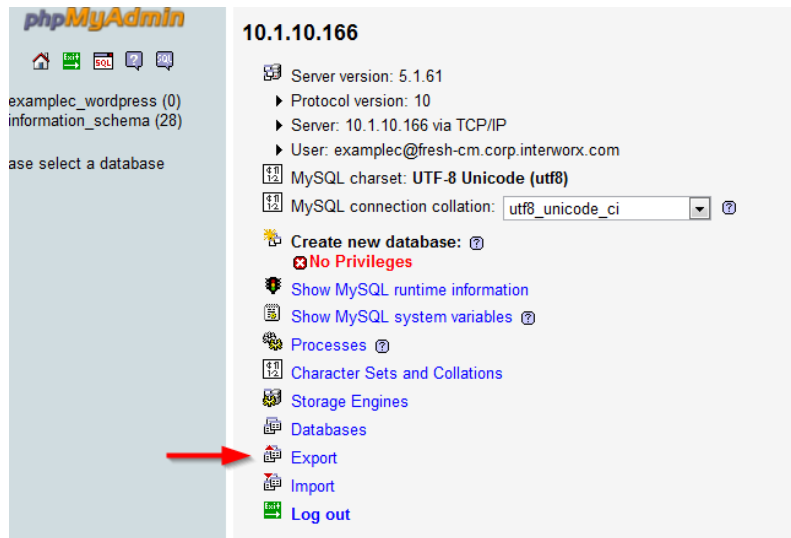


Figure 6.6: phpMyAdmin's Export Link

you add a new user to your web application, you need to do INSERTs on 5 different tables, those are each independent INSERTs that occur independent of each other. If you happen to do a dump while the INSERTs are occurring, you might not get all the data needed for that new user. Transactions are a way of grouping queries together so they occur as a single functional unit - and any changes that are going to be done to the database have to wait until the dump has completed to occur. This ensures you don't get any non-consistent states in your dump.

Disable foreign key checks also only matters if you are using InnoDB tables. Foreign keys are a way of telling the database that one column of one table references the exact same data in another table. For example a user table and a purchase record might have the userid of the user who purchased something in the purchase record table. The foreign key tells the database that userid in purchase record refers to userid in user table, and that if for some reason the user is deleted, that the purchase records relating to that user either need to be deleted or changed to a different userid. Disabling the foreign key checks would make it so when re-creating the database, foreign keys aren't enforced.

Add DROP DATABASE will add a DROP DATABASE for the database being re-created at the beginning of the import. This guarantees that any bad data is wiped away before the import process begins. This also guarantees that any data you have in your database currently is gone, so use with caution.

Add DROP TABLE / VIEW / PROCEDURE / FUNCTION is similar to Add DROP DATABASE. This will delete each table being re-created before it is imported again.

Add IF NOT EXISTS will only re-create tables if they don't exist already. This might be helpful if you are trying to make a backup that isn't destructive to the current data set.

Add AUTO_INCREMENT value is useful if you have columns which rely on auto_increment (many id number columns use this). It's probably best to have this on.

Enclose table and field names with backquotes ensures that names are properly enclosed in backquotes - this isn't 100% necessary but it explicitly defines a table and field name to the SQL system which is important if your column or field name is a MySQL reserved word.¹

Add CREATE PROCEDURE / FUNCTION any procedures or functions you have will attempt to be recreated. Procedures and functions allow you to push some of the data manipulation tasks to the database management system instead of having PHP or whatever web language handle it.

¹Reserved words can be seen here: <http://dev.mysql.com/doc/refman/5.5/en/reserved-words.html#table-reserved-words-5.5.27>

[Databases](#)
[SQL](#)
[Status](#)
[Variables](#)
[Charsets](#)
[Engines](#)
[Processes](#)
[Export](#)
[Import](#)

View dump (schema) of databases

Export

Select All / Unselect All

- examplec_wordpress
- information_schema

CSV
 CSV for MS Excel
 Microsoft Excel 2000
 Microsoft Word 2000
 LaTeX
 Open Document Spreadsheet
 Open Document Text
 PDF
 SQL
 YAML

Options

Add custom comment into header (\n splits lines)

Enclose export in a transaction
 Disable foreign key checks
 SQL compatibility mode: NONE

Database export options

Add DROP DATABASE

Structure

Add DROP TABLE / VIEW / PROCEDURE / FUNCTION
 Add IF NOT EXISTS
 Add AUTO_INCREMENT value
 Enclose table and field names with backquotes
 Add CREATE PROCEDURE / FUNCTION

Add into comments

Creation/Update/Check dates

Data

Complete inserts
 Extended inserts
 Maximal length of created query: 50000
 Use delayed inserts
 Use ignore inserts
 Use hexadecimal for BLOB
 Export type: INSERT

Save as file

File name template (1): __SERVER__ (remember template)

Compression: None "zipped" "gzipped"

i (1) This value is interpreted using `strftime`, so you can use time formatting strings. Additionally the following transformations will happen: __SERVER__

Figure 6.7: phpMyAdmin's export settings screen

Creation/Update/Check dates will add to every table when it created/updated/checked in the comments surrounding it.

Complete inserts adds the column names to the SQL dump. This parameter improves the readability and reliability of the dump. Adding the column names increases the size of the dump, but when combined with Extended inserts it's negligible.

Extended inserts combines multiple rows of data into a single INSERT query. This will significantly decrease filesize for large SQL dumps, increases the INSERT speed when imported, and is generally recommended.

Maximal length of created query the maximum length in characters of an extended query

Use delayed inserts will increase the speed of the import, which might be helpful if you are going to use this SQL file on a very busy server. Normally an insert will return "OK" when the data base has safely stored the data. A delayed insert will return OK immediately, and then queue the insert for when there is a free thread available to service the insert. On busy MySQL servers, you may want this to avoid having a giant import take a long time while it waits for free threads to service each insert.

Use ignore inserts means that errors that occur while executing the INSERT statement are treated as warnings instead. For example, without IGNORE, a row that duplicates an existing UNIQUE index or PRIMARY KEY value in the table causes a duplicate-key error and the statement is aborted. With IGNORE, the row still is not inserted, but no error is issued.

Use hexadecimal for BLOB will use hexadecimal² representation for BLOB (Binary Large Object) columns. These are typically used when storing files, images, data in the database. This is recommended if you have blob columns.

Export type controls how data is put into tables. INSERT is your generic table insert - it will fail if the primary key of the row you are inserting matches an existing primary key. UPDATE will only add the data if there is an existing row with a primary key that matches the one in the row you are updating. REPLACE behaves a lot like INSERT, but instead will delete a row if its primary key matches the row you are REPLACE-ing.

File Name Template will control what the filename will be when downloading the dump from the server. Keep in mind that you can always save the SQL file as what ever you choose from your browser.

Compression controls whether the dump will be compressed before making it available for you to download. This may be desirable if the file is extremely large (over 100 MB).

The default selections should work fine in most cases, but you may wish to tweak the options if you are trying to produce an SQL backup that is non-destructive to existing data.

6.2 Backup on Command Line

If you are an advanced user who wants a bit more control over the backup process on your system, it might be better to do backups via the command-line instead of the interface. While the interface will handle most needs just fine, you may want to increase the rate of how often backups are done, or elect to go around InterWorx's backup system entirely to just have raw MySQL dumps as seen in section [6.1.2.3](#).

6.2.1 Using the command-line backup script

The command line backup script is located at /home/interworx/bin/backup.pex. This is actually the script that gets executed by the panel on your behalf when doing any sort of backup operation. The command line backup interface is less user-friendly, but you have a lot more options at your disposal. Using the command line interface is most useful when you want to automate (e.g., via CRON) the creation of SiteWorx account backups.

In order to use the command-line script, SSH into the server and su to the iworx system user. You may need to switch to the root user first (if you're not already logged in as the root user). Alternatively, you can also switch to

²<http://en.wikipedia.org/wiki/Hexadecimal>

<i>Parameter</i>	<i>Description</i>
<code>--domains [SiteWorx domain list]</code>	Space-separated list of SiteWorx account domains to backup. Simple regular expressions are also allowed here (see examples below).
<code>--domains all</code>	Backup ALL domains on the server
<code>--backup-options db</code>	Backup databases only
<code>--output-dir [path]</code>	Optional - set where you want the final backup file to reside
<code>--tmp-dir [path]</code>	Optional - When the backup is being created, InterWorx needs to create temporary files somewhere. By default it's /tmp. You can set the tmp dir with this parameter
<code>--xfer-method [method]</code>	Optional (requires <code>--xfer-ini</code>) - If you want IWorx to send the backup(s) to another server, you can specify either ftp or scp as the method.
<code>--xfer-ini [path/to/xfer.ini]</code>	Optional - In order to use remote transfer, you need to specify a file with transfer settings. This is described later.
<code>--reseller-id [reseller-id]</code>	Optional - Limits the list of possible backup domains to the domains belonging to the reseller-id. If <code>--domains</code> parameter is not set, but <code>--reseller-id</code> is set, all domains under the given reseller are backed up. If <code>--domains</code> parameter is also set, the list will only match domains belonging to the given reseller.
<code>--compression [1-9]</code>	Optional - Set the compression level for the final backup file. This option is identical to the gzip compression parameter, where 1 is the "quickest" and 9 is the "slowest". If not set, it defaults to the gzip default compression level.
<code>--quiet</code>	Optional - Causes the script to run silently, and not print any text to the screen. Useful for cron jobs.
<code>--email [email]</code>	Optional - Email address to send backup results to.
<code>--filename-format [format-string]</code>	Optional - Sets the format of the final backup filename given the format-string provided. If this option is omitted, the default format used is set in the <code>iworx.ini</code> , under the <code>[iworx.backup][filename_format]</code> setting (default <code>%D-%t-%b.%d.%Y-%H.%M.%S</code>). %D = domain name, %T = unix timestamp, %U = unix username, %R = reseller id, %t = backup type (full or partial), %H = hour, %M = minute, %S = second, %m = month (1..12), %d = day of month (1..31), %a = 3-letter day of week (Sun..Sat), %Y = 4-digit year"

Table 6.1: backup.pex Available Options

the `iworx` user (once you have enabled the `iworx` user account and set a shell). The options related to backing up a database are presented in table 6.1. Using this table, you should be able to create a backup command that can be either scripted or added to crontab for automated backup. Here's an example backup command that you could add to the root user or `iworx` user's crontab³:

```
~iworx/bin/backup.pex --domains all \
--backup-options db \
--output-dir /var/db-backups/ \
--quiet \
--email admin@webhost.com
```

If you want to use the remote transfer options of the backup system, you need to remove the `--output-dir` option, and add `--xfer-method [ftp|scp]` and `--xfer-ini [path/to/xfer.ini]`. The `xfer.ini` is a small text file somewhere on your filesystem that looks like the following for SCP transfers:

```
[ scp ]
username="unix-user"
password="your password"
```

³Backslashes `\` at the end of the line are used to allow the command to span multiple lines in the BASH shell

```
host name="host name.domain.tld"
remotepath = "~/yourbackupdir/"
port="22"
```

Alternatively, if using FTP to transfer the final backup, you want an xfer.ini that looks like this:

```
[ ftp ]
username="ftp-user"
password="your password"
host name="ftp.domain.tld"
remotepath = "/yourbackupdir/"
port="21"
```

6.2.2 Using mysqldump

You may also elect to use the mysql-provided utility to backup your databases, `mysqldump`. This section will in general be brief since the official MySQL documentation already covers this utility well. It should be known that ultimately when doing a backup of a database using the InterWorx backup system, this is the utility that is called to generate the database backup. In order to get login credentials for the database, you can either use the MySQL username and password that has access to the database as specified in SiteWorx, or you can use the root MySQL user. If you missed on how to get the root MySQL password, it needs to be set manually in the control panel as described in section 2.2.1. Here's an example of how to use `mysqldump`:

```
mysqldump -S /var/lib/mysql/mysql.sock -u root --all-databases -p
```

Which would do a full dump of every database on the server. There's also:

```
mysqldump -S /var/lib/mysql/mysql.sock -u root -p [database name]
```

Which would allow you to dump a specific database. Note that by default, `mysqldump` dumps the SQL backup to screen. In order to capture it, you will want to redirect STDOUT to a file like so:

```
mysqldump -S /var/lib/mysql/mysql.sock -u root -p [database name] > /root/backup.sql
```

You will still see the prompt for the user's password and be able to enter it, but the output from the dump will appear in the file after the bash redirection symbol `>`. If the file is large you may wish to gzip it to reduce its size.

6.3 Checking and Repairing corrupted data

Corruption is unfortunately a reality of dealing with database management systems in hosting environments. Luckily, MySQL and the under-lying layers of the filesystem are designed to be able to recover from this sort trouble often by trying to keep a history of all data manipulation stored safely on magnetic or flash media. That way, a loss of power or hard crash allows the software to recover based on the stored list of operations. In addition, corruption can occur when the partition MySQL data is stored on runs out of free space. MySQL may only be able to write part of the data it needs to write for an INSERT or UPDATE to disk before the filesystem cuts off write permission. This means MySQL is stuck running with data in memory that it can't flush to disk, which can lead to corruption. It's important to ensure your disk partitions always have enough space. InterWorx has built in low-space detection and will email the NodeWorx administrator when it detects low on disk space, as well as notify with a warning when logging into NodeWorx.

Often in MySQL, corruption doesn't rear its head until you attempt a backup. Since a backup typically requires stepping through an entire database and pulling out data, it will fail if any sort of corruption is detected. This also means automated backups will not be generating backups until the corruption is fixed. That's why it's important to stay on top of your backup status messages being sent from the system to your email so that you don't risk getting caught with an old state of a SiteWorx account.

6.3.1 Checking for corruption

You can check for corruption using the `mysqlcheck` utility, included by default with most MySQL packages. The MySQL check can give you a definitive report on the status of a table. Again, check section 2.2.1 if you are not aware of how to get the root MySQL password. Then you can run: `mysqlcheck -S /var/lib/mysql/mysql.sock -u root -p --all-databases` to check all databases for corruption. The output for `mysqlcheck` looks like:

```
examplec_WordPress.test          OK
mysql.columns_priv              OK
mysql.db                        OK
mysql.event                     OK
mysql.func                      OK
mysql.general_log
Error      : You can't use locks with log tables.
status    : OK
mysql.help_category             OK
mysql.help_keyword              OK
mysql.help_relation             OK
mysql.help_topic                OK
mysql.host                      OK
mysql.ndb_binlog_index          OK
mysql.plugin                    OK
mysql.proc                      OK
mysql.procs_priv                OK
mysql.servers                   OK
mysql.slow_log
Error      : You can't use locks with log tables.
status    : OK
mysql.tables_priv               OK
mysql.time_zone                 OK
mysql.time_zone_leap_second     OK
mysql.time_zone_name            OK
mysql.time_zone_transition      OK
mysql.time_zone_transition_type OK
mysql.user                      OK
```

On the other hand if there's corruption you might see that one of your tables has a corrupted message:

```
examplec_WordPress.test
error      : Size of datafile is: 162          Should be: 220
error      : Corrupt
mysql.columns_priv              OK
mysql.db                        OK
mysql.event                     OK
mysql.func                      OK
mysql.general_log
.....
```

Occasionally, your corruption will be unrecoverable and you will have to repair the database by importing a backed up state of it. On the other hand, occasionally you can try to repair the database using MySQL's automatic repair capabilities with option `--auto-repair` appended to the `mysqlcheck` command. This will cause MySQL to go through and try and repair the instances of corruption it finds. *Keep in mind that repair is not a guaranteed "safe" operation - before attempting the repair you should try to preserve the current database state. One technique is to stop the MySQL server and make a copy of the database's directory in `/var/lib/mysql` before starting the server backup to attempt the repair operation.*

Chapter 7

Migrating a SiteWorx account to a different MySQL server

When using MySQL remote servers, or additionally when your server scales to the point that your MySQL server is becoming overwhelmed by some of the larger, busier clients, you may elect to move some SiteWorx accounts around. This is generally not recommended because of possible discrepancies between MySQL versions, and other possible “issues” that crop up when you are moving MySQL data around willy-nilly. Yet when push comes to shove, it is possible to move a SiteWorx account’s MySQL data and re-map that account to another MySQL server.

Using `~iworx/bin/iworxdb-transfer.pex`, you can migrate the data AND re-map a SiteWorx account’s assigned database. You need to specify at minimum a SiteWorx account (by master domain name) and a new server nickname (as discussed in section 3.2) to migrate MySQL data to and re-assign the account. The options available for the script are as follows:

- `--domain=[SiteWorx domain]` sets the SiteWorx domain that is going to have data transferred and it’s MySQL server moved.
- `--new-server-nickname=[MySQL server nick]` is the new MySQL server nickname to move the SiteWorx account to. You are to refer to the MySQL server’s nickname as shown in NodeWorx under System Services > MySQL Server > Remote Servers.
- `--force` will drop (i.e. delete) any databases/tables that exist on the target new MySQL server if they match in name before migrating data over. This option should be used with care. Ensure that there aren’t going to be any collisions of databases or tables prior to attempting the migration.
- `--dont-transfer-data` will not perform any data transfer, it will just do a remapping of the SiteWorx account’s assigned MySQL server. Thus, databases and users will disappear from that SiteWorx account until they are moved over manually.

For example,

```
~iworx/bin/iworxdb-transfer.pex --domain=example.com \  
--new-server-nickname=localhost
```

will transfer a SiteWorx account with master domain `example.com` back to the `localhost` MySQL server. You can also add `--dont-transfer-data` to this command, and move the data/users manually using the backup techniques discussed in section 6.2.2, followed with an import in phpMyAdmin (once users are created), or an import from command-line using the `mysql` command-line client. An import with the command-line `mysql` client would look like:

```
# mysql -S /var/lib/mysql/mysql.sock -u [SiteWorx MySQL Database User] -p  
Enter password: *****  
mysql> source /path/to/mysqldump.sql
```

Chapter 8

Odds and Ends

8.1 Command-Line Reference

`service mysqld [start|stop|restart|status] or /etc/init.d/mysqld [start|stop|restart|status]`
Start/Stop/Restart/Check status of the MySQL server daemon

`chkconfig --levels [12345] mysqld [on|off]` Enable/disable mysqld on boot based on linux init run level. Runlevels 3,4,5 are typically considered the “live server” run levels and thus you typically want MySQL to be enabled on those levels.

`~iworx/bin/backup.pex` This is the script which allows you to generate SiteWorx account backups (including MySQL-only partial backups) from command-line. See section 6.2.1 for more details.

`~iworx/bin/iworxdb-transfer.pex` This script allows you to migrate a SiteWorx account’s mysql data from one MySQL server to another, as well as perform a remapping of the current assigned MySQL server to a new one. See chapter 7.

`grep rootdsn ~iworx/iworx.ini` This command is used to extract the iworx MySQL user password from the InterWorx configuration file. You should typically strive to use the root password yourself, but this can be handy when InterWorx has error messages about not being able to connect to the MySQL database. You can check to see if perhaps there’s an issue with the root password.

`mysql` The command-line mysql client.

`mysqldump` Will dump a database or databases of your choice to screen in the form of SQL queries. These queries can be issued to the server to “recreate” the state of the database on demand. Used for generating database backups.

`mysqlcheck` Used to check the integrity of the database files on the filesystem. Can also be used to repair the databases automatically if any corruption is found.

`/etc/my.cnf` The configuration file for the MySQL server

`/var/lib/mysql` Where the actual data for all your MySQL databases “lives” on the file system.

8.2 How MySQL users and databases are mapped to SiteWorx accounts

When creating a MySQL database or user in SiteWorx, InterWorx automatically appends the unix user name of the SiteWorx account to every database/user in order to know which database/user belongs to whatever SiteWorx account. This is why it's critical to not modify the phrase or text in front of the underscore '_' character in any username or database from outside InterWorx. You can mistakenly un-map a user or database from the SiteWorx account and deny the user the ability to control that user or database from inside their SiteWorx panel.

This is also how InterWorx is able to create temporary users on the fly when a SiteWorx user visits phpMyAdmin and is able to see all the databases across all their users. The MySQL user is created with a long, random string password and is only permitted to login from localhost (127.0.0.1) such that this is exclusively a phpMyAdmin-only MySQL user. The user is GRANTED all privileges on the databases that are pre-pended with the SiteWorx unix user name.

8.3 Clustering Considerations

When using clustering, you have to keep in mind that your web application is going to be running and served from multiple servers, not just one. That is why it is critical that you pay attention to how you have your MySQL user permissions setup when creating users for the databases backing the web applications. The easiest thing to do is simply allow the MySQL user to log in from '%', which is the wildcard character that means a user can log in from any host. Alternatively, you can set it up so that you have one user/host combination per server (with identical passwords, usernames, and permissions) that allows each server in the cluster access to that database. This is often not recommended if the SiteWorx user is not aware of the cluster's infrastructure. An alternative is to setup a remote MySQL server on a private network, have every server on the cluster have access to that private network, and create the user with the '%' wildcard host name. This ensures that people from the outside would not be able to get access to your MySQL server directly while at the same time ensuring that every server in the cluster is able to log in as the user for the web app.